# Projek Ilmiah Tahap Akhir II
# WXEX 3181
# Sales, Purchases & Stock System
# WQT 000096
# Mok Chee Hoe

**Table of Contents**

# Introduction

# 1.0 INTRODUCTION

## 1.1 Project Overview

Like any other general accounting software, Sales & Purchases Recording System is designed in such a way that it can fit into use of different types of organizations. It consists of an accounting database which stores all information regarding the organization's trading status.

Since it should fit to any types of organization, Sales & Purchases Recording System has the capability of allowing multiple users to create their own database files and gain access to them. The software is one of dynamic file allocation in which users can specify and create their own database files. Records inside those files are also of preference of the users.

As the software allows multiple users to create their own files, security becomes the main issue. Sales & Purchases Recording System tackles this problem by implementing password protection in which only authorized users can gain access to their files. This is to ensure that the files are kept safe and secure.

The software provide features to generate the following accounting modules: Sales Transaction Journal, Purchases Transaction Journal and Inventory Management Modules. All those modules will be automatically generated after the users have closed their accounts. These reports are certainly based on the information of accounts keyed in by users.

The software is one with Graphical User Interface (GUI) and is Windows-based application. It provides good interface design with visual clues to help users better understand the software.

**Objectives of the software development project**

- To enhance the efficiency of the business transaction recording processes.
- To improve the accuracy of the flows of the processes of recording system as a whole.

**Procedures of development process**

- To conduct a study on the needs of the organization. To produce a summary report of requirement.
- To conduct a study on the pros and cons of the existing system. To prepare a summary report of the existing system.
- To find out the scope and limitations of the existing system. To prepare the scope and constraints report.
- To conduct an interviewing with the management/users of the organization. To ensure that the users' problems and requirements are fully understood. To produce a summary report on the problems faced and the new specifications required by the organization.
- To provide solutions and alternatives to solve the identified problems. To produce the recommendations report.
- To combine all the accomplished reports. To produce the preliminary investigation report.
- Define completely the policies, terms and specifications of the project system.
- Design and development.
- Perform system and user acceptance tests.
- Evaluation and continuously maintenances and improvement.

## 1.2 Major Software Functions

**Features of the software**

Sales & Purchases Recording Software is designed in such a way that it will satisfy users from different types of organization. It provides the following features, which are general to all organizations' accounting activity:

1. **Sales Transaction Recording Module**

   Sales Journal is a specific journal used to record down all sales of merchandise on account.

2. **Purchases Transaction Recording Module**

   Similar to sales journal, the only difference is purchase journal is used to record down all purchases of merchandise on account.

3. **Receipt Journal**

   Receipt Journal is used to record down all collections of merchandise purchased by customer.

4. **Payment Journal**

   Similar to Receipt Journal, the only difference is that Payment Journal will record down all payment of merchandise purchased by the organization instead of the customer.

5. **Inventory Management Module**

   This module is used to provide a monitoring on the movement, status and location of every item in the stock of the organization.

## 1.3 Performance/Behavior Issues

Sales & Purchases Recording System includes all of the five basic accounting modules described in point 1 to 5 above. It is itself a generic accounting software whereby it will be able to fit different types of organization. The features inside the software are designed in such a way that it could adapt itself to global market. One special feature of the software is that it allows multiple users to use the same software on different types of accounting activities.

The software has a main navigation screen which will then allow users to gain access to different types of journal. In the main screen also, users can gain a quick view about the organization's accounting activity.

Every software development should have proper planning. It is to guide the development process towards its goals so the development process will go smoothly. Project planning also means setting out plan to anticipate problems that might be encountered while developing a system. It should cover every single process of the development from specification to the delivery of the system itself. It is an iterative process whereby it would be refined every time the project manager has new information about the project development. With no doubt, project planning is a must in every software development process. It is to help developers identify problems or constraints affecting a project, set specific production timeline (milestone and deliverables) to be achieved and develop plan to tackle any anticipated problem.

## 1.4 Scope and Constraints

**Proposed system**

**Scope:**

1. The accounting package that we are looking forward to develop will be run in client-server environment. It would provide access to multiple users in the workspace and allow input of relevant data on-line.
2. Since security is of high-level importance, therefore the package that we are going to develop will provide strong control whereby level of accessing for each user is based on their authorities. In this case, login name and password will be implemented accordingly.
3. The database will be designed in such away that it meets requirements of different type of industries. It will thus provide the software with flexible capabilities.
4. Reports generated will include most accounting models such as sales journal, purchase journal and other important report for analysis purposes.

**Constraints:**

*1. Within three months time scale*

The package is to be developed in less than 4 (four) months time, resulting in tight schedule to be introduced among team members.

*2. Must use minimum cost*

The software is required to be developed in minimum cost. Limitations in term of cost will more or less become obstacles in developing the software.

*3. The software should follow accounting standards*

Each modules included in the software should meet accounting standards of different types of industries. This means that the software should be general enough in terms of requirements. Therefore, specialization in a particular field should not be introduced.

**Current System**

**Scope:**

1. The company is currently running a DOS-based accounting system. The software uses menu-driven interface to communicate with users.
2. The current system has strong security control whereby users have to key in their login name and correct password in order to gain access to the database. The users' accesses are based on their positions in the company.
3. Reports can be generated either on a monthly or yearly basis. This gives sufficient information to the accounting department. Reports generated are text-based.
4. The software provides back-up for each calendar year. Users can also gain access to back-up files easily.

**Constraints:**

*1. The software does not provide GUI (Graphical User Interface)*

Interface provided in the system is text-based and menu-driven. Users have to explore the functionality of each menu by experimenting on it. Logical conclusion and interpretation is hard to achieve.

*2. Only text-based reports are generated*

Report needed for each accounting module is text-based. Summarized information is somehow not available.

*3. Improper design of database*

Database used in the company is not designed properly. Similar items are not grouped together which results in slow access.

*4. Slow system access*

Access to the database is slow and cumbersome, resulting users to wait for a few minutes to derive information needed. Consequently, jobs are performed slowly.

# Project Estimates

# 2.0 PROJECT ESTIMATES

## 2.1 Project Resources

### 2.1.1 People

This project will require a programmer consultant in order to be given appropriate advices during the each stage of development process. Therefore, the time available for development could be utilized more efficiently. It is also solving the technical problems due to the limitation on the skills and technical knowledge of the developer.

### 2.1.2 Minimal Hardware Requirements

**Development**

Three IBM PC or compatibles with the following configurations

- . Intel Pentium II 333MHz processor
- . 64MB SDRAM
- 600MB Hard disk space
- . Network Connection

**User Client-side**

IBM PC or compatibles with the following configurations

- . Intel Pentium II 333MHz processor
- . 32MB SDRAM
- . 20MB Hard disk space

**User Server-side**

IBM PC or compatibles with the following configurations

- . Intel Pentium II 333MHz processor
- . 64MB SDRAM
- . 500MB Hard disk space

### 2.1.3 Minimal Software Requirements

**Development**

- . Windows 98
- . Windows NT Workstation
- . Microsoft SQL Server
- Borland Delphi

**User Client-side**

- . Windows 98/NT Workstation
- . Microsoft SQL Server

**User Server-side**

- . Windows NT Server
- . Microsoft SQL Server

### 2.1.4 References of Technical and Development Tool Used in the System Development

- Database Concepts and Architecture (refer to Appendix A)
- SQL Server Background (refer to Appendix B)
- Borland Delphi (refer to Appendix C)

### 2.1.6 Surveying And Anaylsis of System Requirement

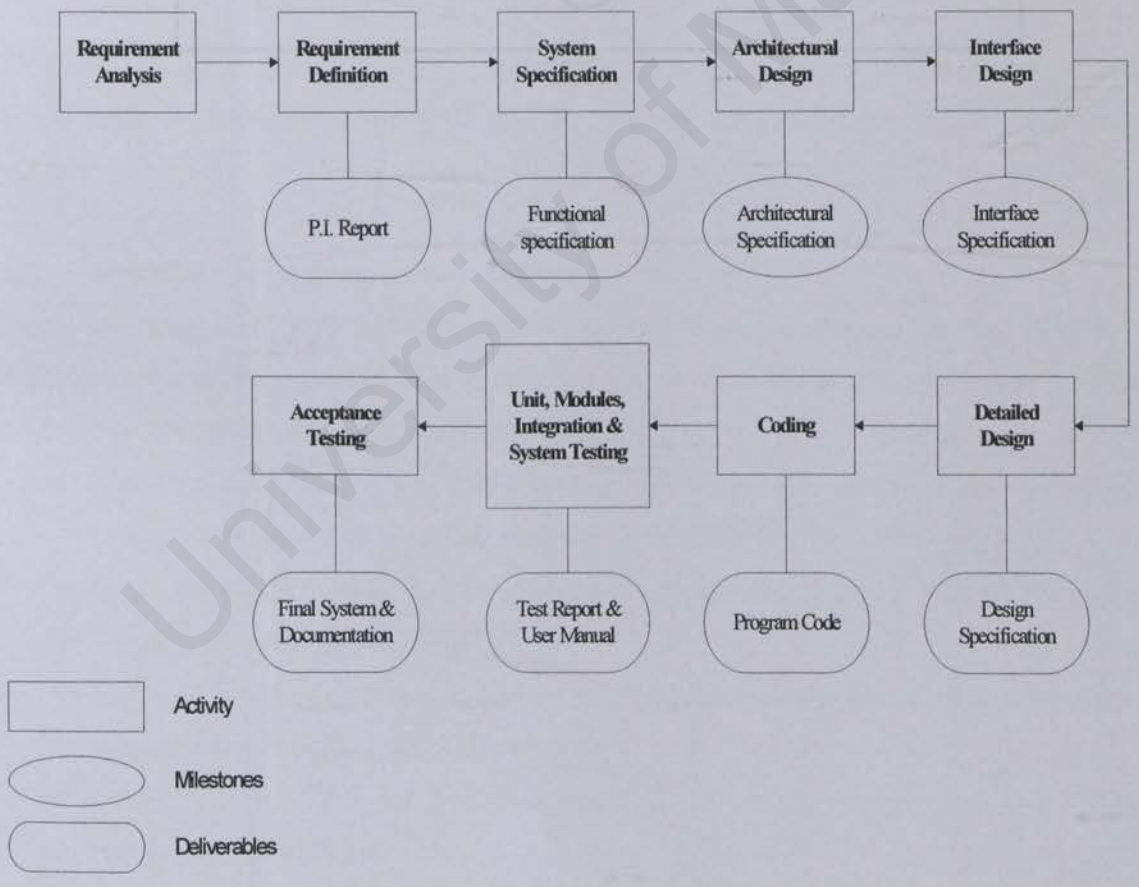
The system development is not only based on my experience in accounting and auditing line but also several reports from interviewing made with the Company management and related staffs for understanding their existing complaints on the existing system and new specification requirements on the new system.

In order to achieve the purposes of understanding, I have designed a number of interview cards for gathering the necessary information for the system development. Details of the Cards could be referred to Appendix D

## 2.3 Project Schedule

| No. | Task Name | Aug | Sep | Oct | Nov | Dec | Jan | Feb |
|---|---|---|---|---|---|---|---|---|
| 1. | Feasibility Study | | | | | | | |
| 2. | System Review | | | | | | | |
| 3. | System Design | | | | | | | |
| 4. | System Specification and Coding | | | | | | | |
| 5. | System Implementation and Testing | | | | | | | |
| 6. | System Evaluation | | | | | | | |
| 7. | Conclusion | | | | | | | |

## 2.3 Project Development Stage



# Risk Management

## 3.0 RISK MANAGEMENT

### 3.1 Scope and intent of RMMM Activities

We want the software to be free of any defects or errors, but it is hard or at times almost impossible to develop a system that is free of any defects. To be safe we would like to have a risk management plan to counter any difficulties that may impact the development or the creation of the software. The goal is to assist the project team in developing a strategy to deal with any risk. For this we will take a look at the possible risks, how to monitor them and how to manage the risk.

For software development to avoid any risk both the developer and client have to work together. Client has to spend time with the developer in the beginning phase of the software development. If client decides to change the software, meaning if client wants to add some more functions into the software or to change the requirement, this will have major effect on the development of the software.

***Critique:*** *Avoiding risk is a noble goal, but risk is inevitable. The real focus of risk management is to understand the risks that are likely to occur and develop plans for dealing with them.*

### 3.2 Risk management organizational role.

Every one associated with the software has responsibility of managing the risk. That is if everyone participated and paid close attention to all the details during the early phase of the software development many risk can be avoided.

- Software development can avoid having risk by double-checking their schedule, product size, estimates regarding costs of the development etc.
- Customer can help avoid risk by providing all necessary software information during the early phase of the development.
- Software development team can avoid risk by getting all the details of the equipment that are provided or are accessible to them.
- Client can avoid risk by making all necessary business changes before initiating request for the software.

### 3.3 Risk Description

This section describes the risks that are likely to be encountered during this project.

***Business Impact Risk:***

This is the risk where concern is that of the not being able to come up or produce the product that has impact on the client's business. If the software produced does not achieve its goals or if it fails to help the business of clients improve in special ways, the software development basically fails.

***Customer Risks:***

This is the risk where concern is client's motivation or willingness in helping the software development team. If the client fails to attend meeting regularly and fails to describe the real need of the business the produces software will not be one that helps the business.

***Development Risks:***

If client fails to provide all the necessary equipment for the development and execution of the software this will cause the software to become a failure. So in other words customer has to be able to provide time and resources for the software development team. If all the requested resources are not provided to the software development team odds for the software development to fail rises greatly.

***Employee Risk:***

This risk is totally dependent on the ability, experience and willingness of the software development team members to create the working product. If the team members are not experience enough to use the application necessary to develop the software it will keep pushing the development dates until it's too late to save the project. If one or more members of the software development team are not putting in all the effort required to finish the project it will cause the project to fail. Employee risk is one of the major risk to consider while designing the software.

***Process Risks:***

Process risk involves risks regarding product quality. If the product developed does not meet the standards set by the customer or the development team it is a failure. This can happen because of the customer's failure to describe the true business need or the failure of the

software development team to understand the project and than to use proper equipment and employees to finish the project.

***Product Size:***

This risk involves misjudgment on behalf of the customer and also the software development team. If the customer fails to provide the proper size of the product that is to be developed it will cause major problems for the completion of the project. If software development team misjudges the size and scope of the project, team may be too small or large for the project thus spending too much money on project or not finishing project at all because of shortage of finances.

***Technology Risk:***

Technology risk involves using technology that already is or is soon to be obsolete in development of the software. Such software will only be functional for short period of time thus taking away resources from the customer. Since the technology changes rapidly these days it is important to pay importance to this risk. If customer request use of software that soon to be obsolete software development team must argue the call and have to pursue customer to keep-up with current technology.

***Anticipated Risk:***

a) ***Software quality may decrease due to tight schedule***

   *Why it might happen:* The software is required to finish in less than three months. This may cause a decrease in software quality, as the team may not have enough time to fully utilize their skill.

b) ***Tight schedule introduced may cause team member to feel less motivated***

   *Why it might happen:* Restless schedule given to team members will cause them feeling tired and exhausted. Thus, motivation among team members will decrease. This will possibly affect the quality of the finiHed product also.

c) ***Client-server environment in software may not be fully met***

   *Why it might happen:* The team has only little expertise in developing software based on client-server environment.

***Risk Resolutions:***

***a) Fully utilized tools and resources to address tight schedule***

The tight schedule can be addressed by utilizing tools and resources that may help finish certain tasks. One example is to use MS Project to help plan and manage activities.

***i) Self-motivate and make the development process an interesting activity***

This will help motivate the team, as the process itself is interesting and educative. The team also understands that whatever they will learn in this project will help their professions in the future.

***ii) Consult knowledgeable person to overcome obstacles***

Less expertise in client-server environment will be addressed by seeking help from people familiar with it. The team will try its best to incorporate the skill.

***iii) Seek for reusable components***

If several modules or components in the package are still reusable, they will be modified to meet the requirements.

### 3.4 Risk Table

The following table describes the risks associated with the project. The appropriate categories of the risks are also given, as well as probability of each risk and its impact on the development process.

***Probability and Impact for Risk Management***

The following is the sorted version of the above table by probability and impact:

| Category | Risks | Probability | Impact |
|---|---|---|---|
| Programmer Risks | Lack of training and experience | 40% | 1 |
| Process Risk | Low product quality | 35% | 1 |
| Product Size | Where size estimates could be wrong | 30% | 2 |
| Development Risks | Insufficient resources | 30% | 2 |

| | | | |
|---|---|---|---|
| Customer Risk | Customer may fail to participate | 20% | 3 |
| Technology Risk | Obsolete technology | 10% | 2 |
| Business Impact | Product may harm the business | 10% | 3 |

*Table 1 - Risks Table (sorted)*

| **Impact Values** | **Description** |
|---|---|
| 1 | Catastrophic |
| 2 | Critical |
| 3 | Marginal |
| 4 | Negligible |

Above is the table that categorizes the risks involved in software development. It gives brief description of the risk in Risks column and also provides the probability of risk occurring in percentages in Probability column and also the impact of the risk in the Impact column.

The impacts values assigned to the each risk are described in the section below the risk table. It is very convenient way to look at the risk and derive the information of the risk.

***Critique:*** *It is important to provide a reference to the RMMM document (which does exist elsewhere in the project docs.). Even if the RMMM Plan has not as yet been developed, the Project plan should provide an indication that it will be developed and a provisional pointer to it.*

### 3.5 Risk Management

Risk monitoring involves regularly assessing each of the identified risks to decide whether or not that risk is becoming more or less probable and whether the effects of risk have changed. Of course, this cannot usually be observed directly, so we have to look at other factors which give us clues about the risk probability and its effects. These factors are obviously dependent on the types of risk.

Risk monitoring should be a continuous process and, at every management progress review, each of the key risks should be considered separately and discussed by the meeting.

# System Design

## 4.0 System Design

Design is the creative process of transforming the problem into a solution. A design consists of two parts which are the conceptual design and technical design.

A conceptual design tells customer exactly what the system will do, whereas the technical design allows the system builders to understanding the actual hard ware and software need to solve the customer's problems.

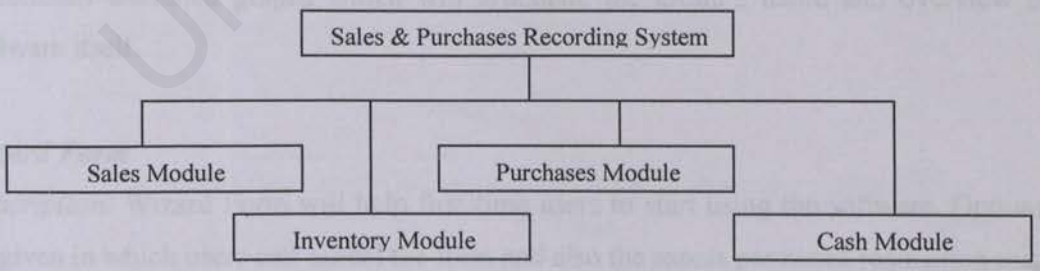
In this chapter, the following designs are discussed:

## 4.1 Architectural Design

The primary objective of architectural design is to develop a modular program structure and represent the control relationship between the modules. In the Sales & Purchases Recording System is mainly divided into 4 modules which are:

Sales Module,
Purchases Module,
Inventory Module and
Cash Module.

Sales Module is mainly used to deal with any sales transaction and also the detail information of customer, whereas Purchases Module is used to deal with purchases transaction and the detail information of supplier. The Inventory Module is used to keep track the information and status of products on hand as well as in which location they put. Cash Module is used to manage the outflows and inflows of money in the company whether in term of cash or cheque.

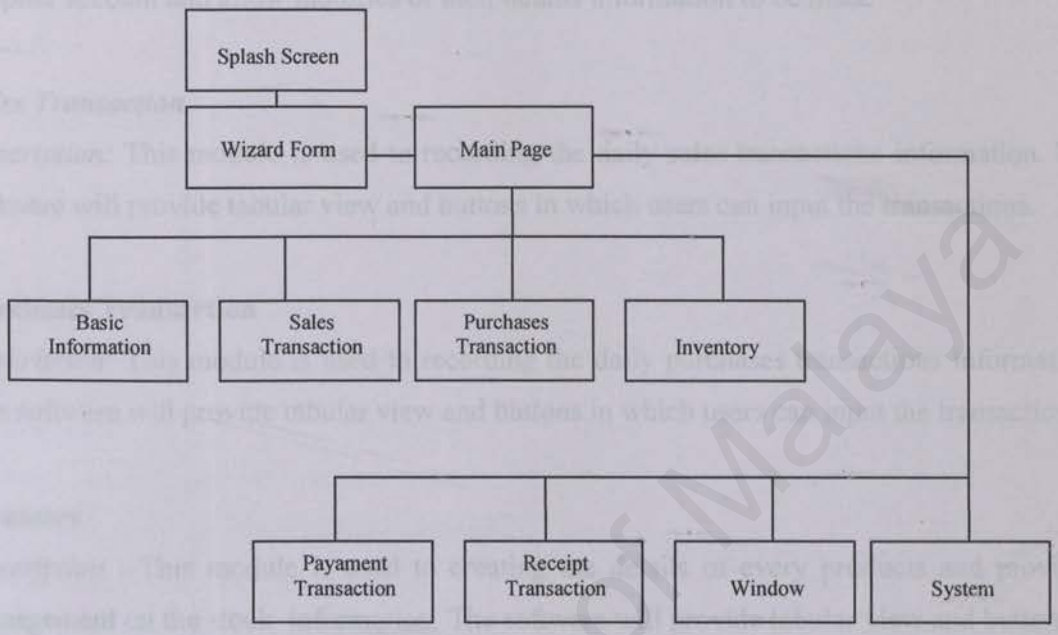



**Sales & Purchases Recording System Architetural Design**

## 4.2 System Functionality Design

However, the conceptual design of the System is divided into 4 modules but it was not divided into the same structure as in the Functionality Design, this due to the need of adaptation to user interface design and system maintenance purposes.

**The Structure Of System Functionality Design**




### 4.2.1 Modules Description

***Splash Screen***

*Description*: Splash Screen will be the first thing users will see when they start the software. It contains animated graphs which will articulate the group's name and overview of the software itself.

***Wizard Form***

*Description:* Wizard Form will help first-time users to start using the software. Options will be given in which users can cancel the form and also the assess password requisition menu.

***Main Page***

*Description:* The main page provides all the features the software has. It also acts as a link between forms included in the system. Design will be made in such a way in which it will contain visual clues to better communicate with users.

***Basic Information***

*Description:* The basic information is menu for creating company profile, customer and supplier account and allow inquiries of their details information to be made

***Sales Transaction***

*Description:* This module is used to recording the daily sales transactions information. The software will provide tabular view and buttons in which users can input the transactions.

**Purchases Transaction**

*Description:* This module is used to recording the daily purchases transactions information. The software will provide tabular view and buttons in which users can input the transactions.

***Inventory***

*Description* : This module is used to creating the details of every products and provided management on the stock information. The software will provide tabular view and buttons in which users can input the transactions.

***Payment and Receipt Transaction***

*Description:* To manage the cash inflows and outflows which are related to the sales and purchases activities.

***Window***

*Description:* This is the program and user management modules.

***System***

*Description:* Access to the security system and information about the application.

### 4.3.0 The Functionality of Sub-Modules

For each major functional module, they are constituted by several sub-modules which their features of function are capable be grouped together.

Due to the number of sub-modules are much, hence a control system must be imposed on modules during the design process as well as for the future maintenance purposes.

Illustration :

AA000

AA: indicated the sub-modules nature,
0000: indicated the code of the modules in the same group of AA.

### *4.3.1 Basic Information*

BA010 Region Number Maintenance
BA020 Customer Type Code Maintenance
BA030 Supplier Type Code Maintenance
BA040 Inventory Change Code Maintenance
BA050 Warehouse Code Maintenance
BA060 Product Type Maintenance
BA070 Bank Information Maintenance

BA110 Sales Man Maintenance
BA120 Customer Account Maintenance
BA130 Supplier Account Maintenance
BA140 Product Maintenance

BA210 Region Code Report
BA220 Customer Type Report
BA230 Supplier Type Report
BA240 Inventory Change Report
BA250 Warehouse Report
BA260 Bank Information Report
BA270 Sales Man Report

BA280 Customer Account Report

BA290 Supplier Account Report

BA300 Product Type Report

### *4.3.2 Purchases Transaction*

PR110 Purchases Invoice Maintenance

PR210 Purchases Item List

PR220 Consignment Item List

PR230 Return Outward List

PR240 Unreturned Consignment List

PR250 Purchase Item Analysis Report

PR260 Purchase Item Ranking List

PR270 Annual Purchases Analysis

### *4.3.3 Sales Transaction*

DL110 Sales Invoice Maintenance

DL210 Sales Item List

DL220 Consignment Item List

DL230 Return Inward List

DL240 Unreturned Consignment List

DL250 Sales Item Analysis Report

DL260 Sales Item Ranking List

DL270 Annual Sales Analysis

### *4.3.4 Inventory*

IN100 Inventory Beginning Balance

IN110 Inventory Change Record

IN120 Inventory Transfer Record

IM210 Stock Count List

IN220 Product Change Detail List

IN230 Product Change Summary Report

IN240 Inventory Transfer Details List

IN250 Inventory Safety Level Report

IN260 Obsolesced Stock Analysis Report

IN270 Monthly Stock Summary

### *4.3.5 Payment Transaction*

AP110 Payment

AP210 Daily Payment Report

AP220 Supplier Statement

AP230 Payable Analysis Report

AP240 Advance Payment List

AP250 Supplier Aging Report

AP260 Overdue Payment List

AP270 Payment Status Report

### *4.3.6 Receipt Transaction*

AR110 Receipt

AR210 Daily Receipt Report

AR220 Debtor Statement

AR230 Receivable Analysis Report

AR240 Advance Receipt List

AR250 Debtor Aging Report

AR260 Overdue Receipt List

AR270 Payment Status Report

AR280 Credit Limit Analysis Report

### 4.3.6 *Window*

SY110 User Maintenance

SY120 User Authorities Maintenance

SY130 Program Access Maintenance

SY140 Change of Password

SY150 Company Profile

### *4.3.8 System*

SY210 User Profile

WQT 00096

SY220 Register Code

SY230 User Authorities List

SY240 Program Execution Authorities List

SY990 Month End Closing

## 4.4 Database Design

Database design focuses on the design of the database model that will support the system operations and objectives. In the process of database design, it must concentrate on the data characteristic required to build the database model.

The main activities for this stage are:

Create the conceptual design,

Designing the table structures for the database.

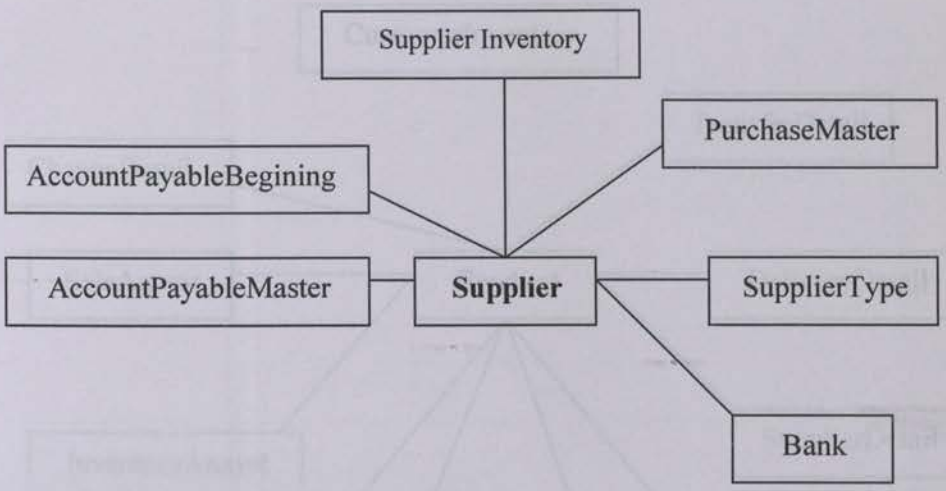
### 4.4.1 Conceptual Design

Data modeling is used to create an abstract database structure that represents real world objects in the most realistic way possible. One of the steps taken in this stage is creating the Entity-Relationship (E-R) Model. The E-R Model is a tool that is used to

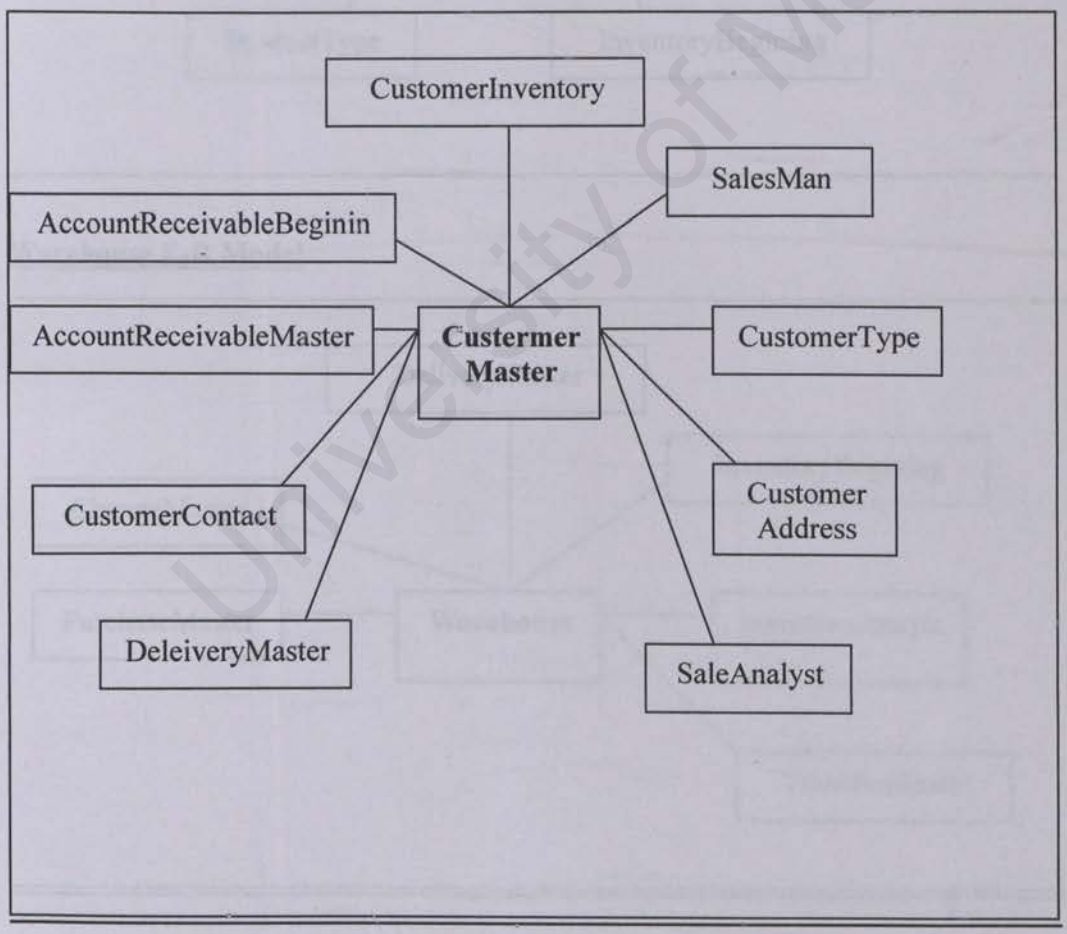
- translated different views of data to fit into a common framework,
- define data processing and constraint requirements to help to meet the different views.
- help implement the database.

The following diagrams are the conceptual design for the database of the System in term of E-R Model.

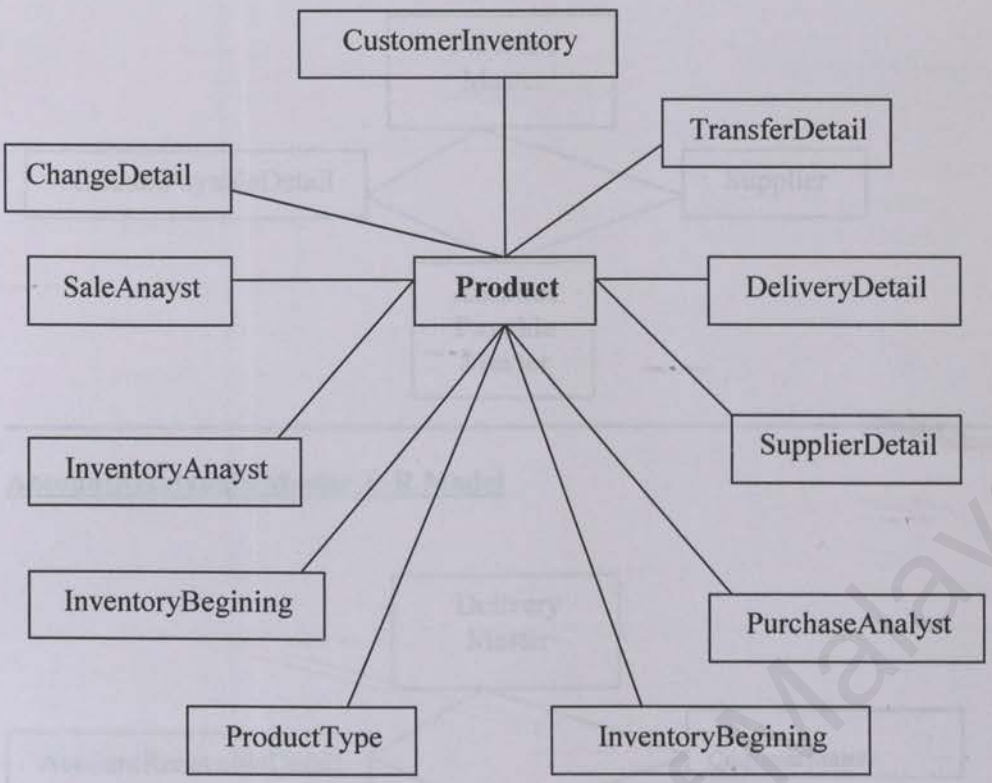
**Supplier E-Rmodel**



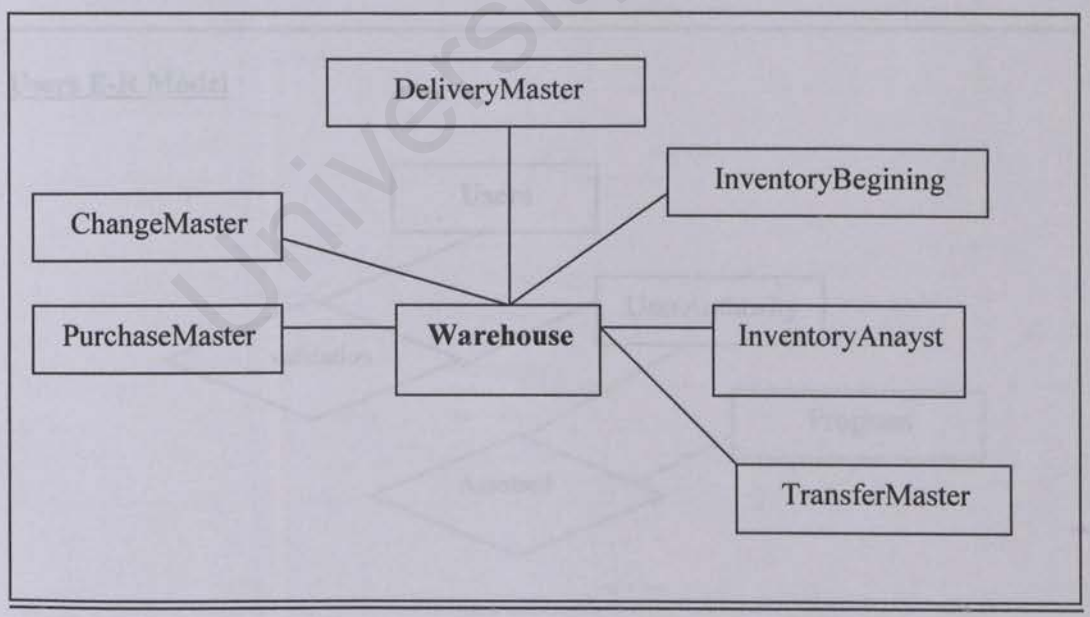

**Custermer Master E-R Model**



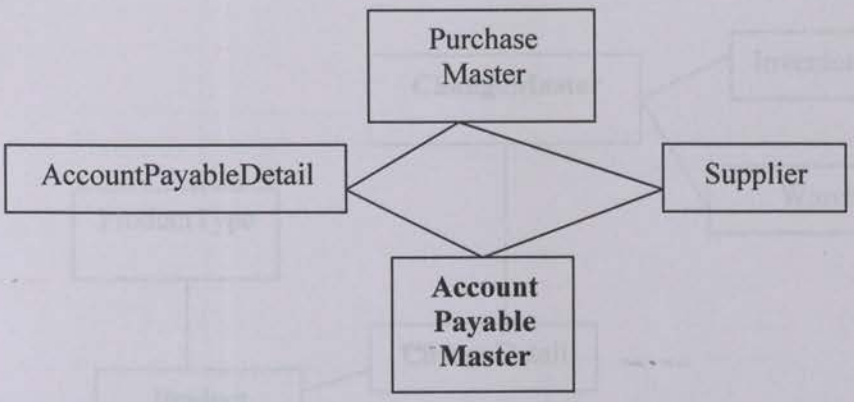
**Product E-R Model**




**Warehouse E-R Model**

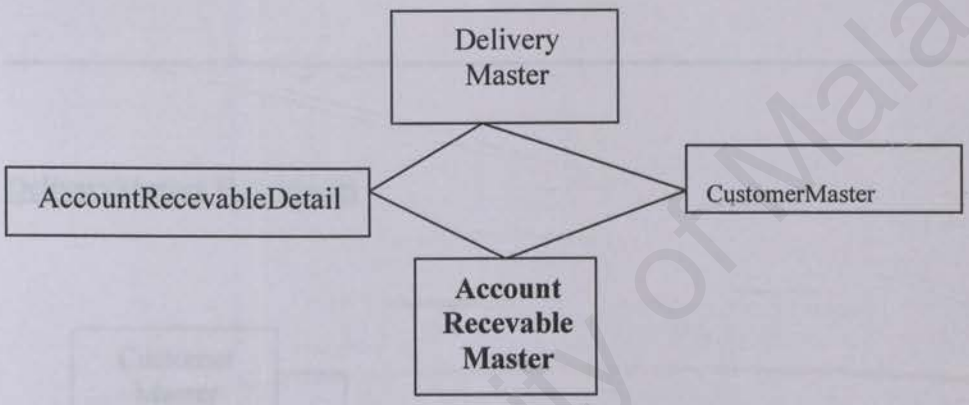


**AccountPayableMaster E-R Model**



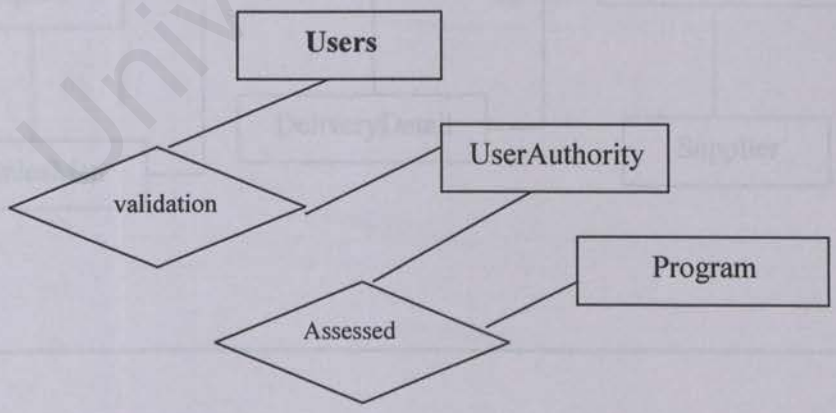

---

**AccountReceivanbeMaster E-R Model**



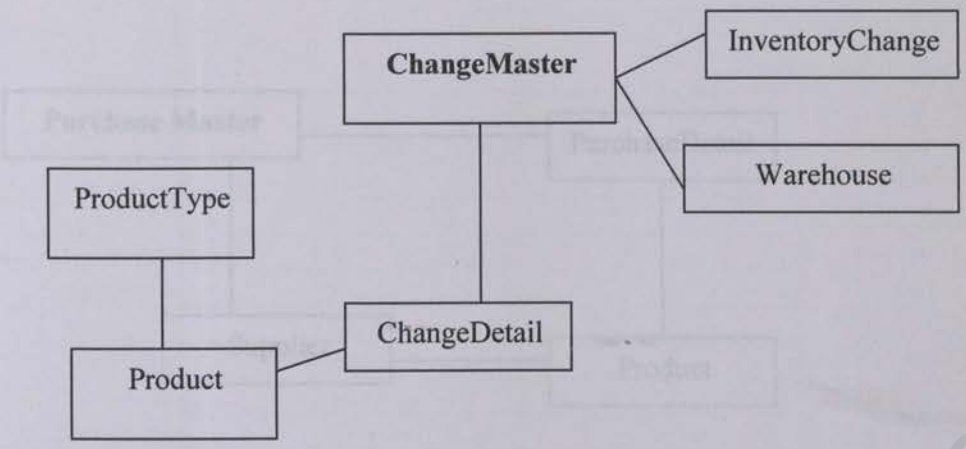

---

**Users E-R Model**



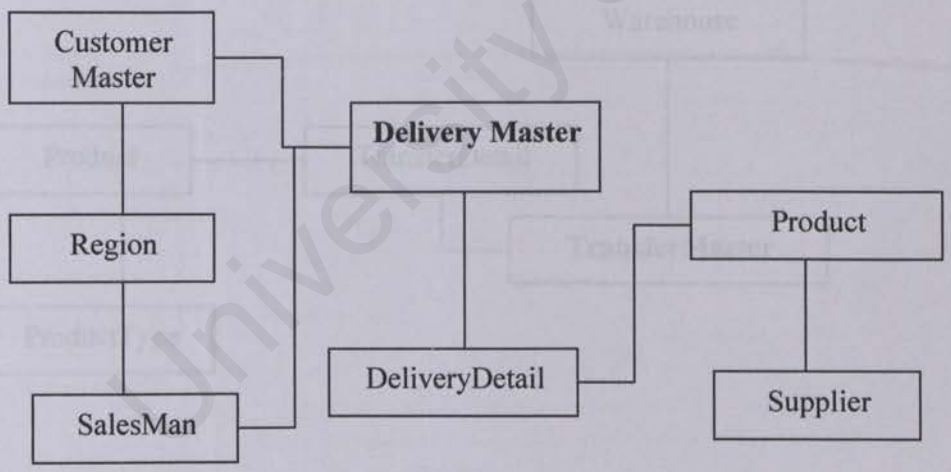

---

**ChangeMaster E-R Model**



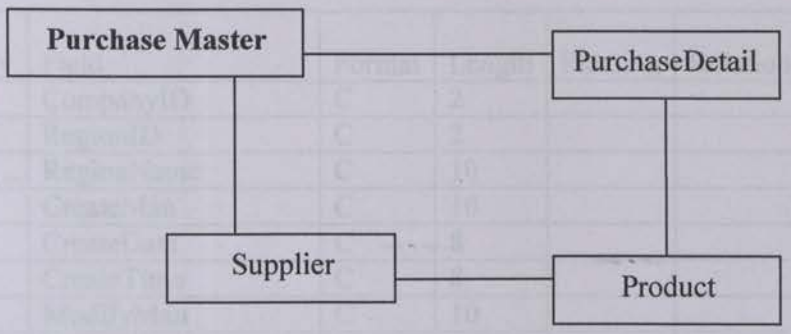

---

**DeliveryMaster E-R Model**



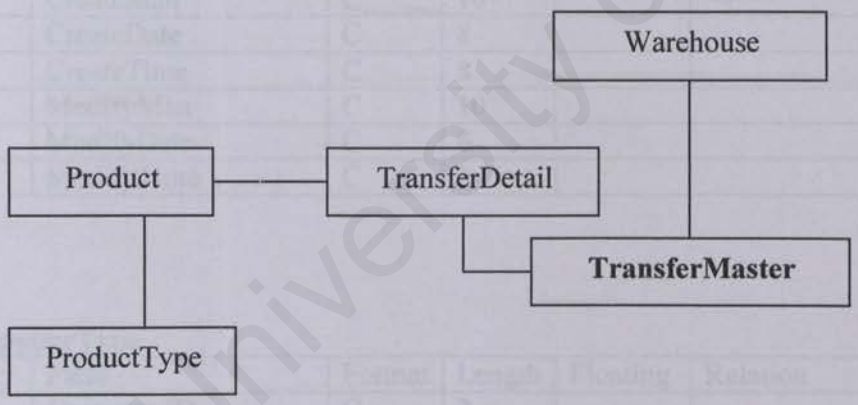

---

**Purchase Master E-R Model**




---

**TransferMaster E-R Model**




---

### 4.4.2 Table Structure in the Database

The tables involved in the Sales & Purchases recording System database are:

**Region**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | RegionID | C | 2 | | |
| | RegionName | C | 10 | | |
| | CreateMan | C | 10 | | |
| | CreateDate | C | 8 | | |
| | CreateTime | C | 8 | | |
| | ModifyMan | C | 10 | | |
| | ModifyDate | C | 8 | | |
| | ModifyTime | C | 8 | | |

**CustomerType**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | CustomerTypeID | C | 2 | | |
| | CustomerTypeName | C | 10 | | |
| | CreateMan | C | 10 | | |
| | CreateDate | C | 8 | | |
| | CreateTime | C | 8 | | |
| | ModifyMan | C | 10 | | |
| | ModifyDate | C | 8 | | |
| | ModifyTime | C | 8 | | |

**SupplierType**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | SupplierTypeID | C | 2 | | |
| | SupplierTypeName | C | 10 | | |
| | CreateMan | C | 10 | | |
| | CreateDate | C | 8 | | |
| | CreateTime | C | 8 | | |
| | ModifyMan | C | 10 | | |
| | ModifyDate | C | 8 | | |
| | ModifyTime | C | 8 | | |

**InventoryChange**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | ChangeCode | C | 2 | | |
| | ChangeName | C | 10 | | |
| | AddOrSubtract | | 1 | | |
| | CreateMan | C | 10 | | |
| | CreateDate | C | 8 | | |
| | CreateTime | C | 8 | | |
| | ModifyMan | C | 10 | | |
| | ModifyDate | C | 8 | | |
| | ModifyTime | C | 8 | | |

**Warehouse**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | WarehouseID | C | 4 | | |
| | WarehouseName | C | 10 | | |
| | CreateMan | C | 10 | | |
| | CreateDate | C | 8 | | |
| | CreateTime | C | 8 | | |
| | ModifyMan | C | 10 | | |
| | ModifyDate | C | 8 | | |
| | ModifyTime | C | 8 | | |

**ProductType**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | ProductTyeID | C | 2 | | |
| | ProductTypeName | C | 10 | | |
| | CreateMan | C | 10 | | |
| | CreateDate | C | 8 | | |
| | CreateTime | C | 8 | | |
| | ModifyMan | C | 10 | | |
| | ModifyDate | C | 8 | | |
| | ModifyTime | C | 8 | | |

**Bank**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | BankID | C | 7 | | |
| | BankAttriName | C | 12 | | |
| | BankName | C | 40 | | |
| | CreateMan | C | 10 | | |
| | CreateDate | C | 8 | | |
| | CreateTime | C | 8 | | |
| | ModifyMan | C | 10 | | |
| | ModifyDate | C | 8 | | |
| | ModifyTime | C | 8 | | |

**SalesMan**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | SalesManID | C | 4 | | |
| | SalesManName | C | 15 | | |
| | RegionID | C | 2 | | * |
| | ContactPhone | C | 20 | | |
| | MobilePhone | C | 10 | | |
| | ContactAddres | C | 60 | | |
| | Email | C | 30 | | |
| | CreateMan | C | 10 | | |
| | CreateDate | C | 8 | | |
| | CreateTime | C | 8 | | |
| | ModifyMan | C | 10 | | |
| | ModifyDate | C | 8 | | |
| | ModifyTime | C | 8 | | |

**CustomerMaster**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | CustomerID | C | 8 | | |

| | | | | | |
|---|---|---|---|---|---|
| | CustomerAttribName | C | 10 | | |
| | CustomerName | C | 60 | | |
| | InvoiceNo | C | 8 | | |
| | CustomerTypeID | C | 2 | | * |
| | RegionID | C | 2 | | * |
| | Owner | C | 8 | | |
| | ICID | C | 10 | | |
| | ContactPhone 1 | C | 20 | | |
| | ContactPhone 2 | C | 20 | | |
| | Fax | C | 20 | | |
| | SalesManID | C | 4 | | * |
| | CustomerAddressID | C | 2 | | * |
| | DeliveryAddressID | C | 2 | | * |
| | InvoiceAddressID | C | 2 | | * |
| | InvoiceType | C | 1 | | |
| | PaymentTerm | C | 1 | | |
| | PayDays | D | 2 | 0 | |
| | CustomerPayDate | C | 8 | | |
| | CreditLine | D | 12 | 0 | |
| | CreditBalance | D | 12 | 0 | |
| | CustomerClass | C | 1 | | |
| | LastDeliveryDate | C | 8 | | |
| | Advance | D | 12 | 0 | |
| | CreateMan | C | 10 | | |
| | CreateDate | C | 8 | | |
| | CreateTime | C | 8 | | |
| | ModifyMan | C | 10 | | |
| | ModifyDate | C | 8 | | |
| | ModifyTime | C | 8 | | |

**CustomerContact**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | CustomerID | C | 8 | | |

| | | | | | |
|---|---|---|---|---|---|
| * | ContactID | C | 10 | | |
| | CustomerName | C | 8 | | |
| | Title | C | 20 | | |
| | Phone | C | 20 | | |
| | MobilePhone | C | 10 | | |
| | Email | C | 30 | | |
| | LiasonRedLetterDate | C | 8 | | |
| | RedLetterDateDescription | C | 20 | | |

**CustomerAddress**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | CustomerID | C | 8 | | |
| * | AddressID | C | 2 | | |
| | AddressDescription | C | 20 | | |
| | PostalCode | C | 10 | | |
| | Line1 | C | 30 | | |
| | Line2 | C | 30 | | |
| | Line3 | C | 30 | | |

**SupplierMaster**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | SupplierID | C | 8 | | |
| | SupplierAttribName | C | 10 | | |
| | SupplierName | C | 60 | | |

| | | | | | |
|---|---|---|---|---|---|
| | InvoiceNo | C | 8 | | |
| | SupplierTypeID | C | 2 | | * |
| | Owner | C | 8 | | |
| | ICID | C | 10 | | |
| | ContactPhone 1 | C | 20 | | |
| | ContactPhone 2 | C | 20 | | |
| | Fax | C | 20 | | |
| | ContactName1 | C | 8 | | |
| | ContactNAme2 | C | 8 | | |
| | CompanyAddress | C | 60 | | |
| | DeliveryAddress | C | 60 | | |
| | InvoiceAddress | C | 60 | | |
| | PaymentTerm | C | 1 | | |
| | SupplierClass | C | 1 | | |
| | LastPurchaseDate | C | 8 | | |
| | Prepaid | D | 12 | 0 | |
| | BankID | C | 7 | | * |
| | AccountNo | C | 14 | | |
| | BankAccountName | C | 60 | | |
| | CreateMan | C | 10 | | |
| | CreateDate | C | 8 | | |
| | CreateTime | C | 8 | | |
| | ModifyMan | C | 10 | | |
| | ModifyDate | C | 8 | | |
| | ModifyTime | C | 8 | | |

**Product**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | ProductID | C | 7 | | |
| | ProductName | C | 12 | | |
| | Unit | C | 40 | | |
| | ProductTypeID | C | 2 | | * |
| | SafeStock | D | 12 | 0 | |
| | LastPuchaseDate | C | 8 | | |
| | LastDeliveryDate | C | 8 | | |

| | | | | | |
|---|---|---|---|---|---|
| | SupplierID1 | C | 8 | | * |
| | SupplierID2 | C | 8 | | * |
| | StopSales | C | 1 | | |
| | CreateMan | C | 10 | | |
| | CreateDate | C | 8 | | |
| | CreateTime | C | 8 | | |
| | ModifyMan | C | 10 | | |
| | ModifyDate | C | 8 | | |
| | ModifyTime | C | 8 | | |

**PurchaseMaster**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | PurchaseID | C | 10 | | |
| | PurchaseDate | C | 8 | | |
| | SupplierID | C | 8 | | * |
| | WarehouseID | | 4 | | * |
| | PurchaseProperty | | 1 | | |
| | InvoiceNo | C | 10 | | |
| | SubTotal | D | 12 | 0 | |

| | | | | | |
|---|---|---|---|---|---|
| | ValueAddTax | D | 12 | 0 | |
| | Amount | D | 12 | 0 | |
| | AccountPayable | D | 12 | 0 | |
| | Paid | D | 12 | 0 | |
| | CreateMan | C | 10 | | |
| | CreateDate | C | 8 | | |
| | CreateTime | C | 8 | | |
| | ModifyMan | C | 10 | | |
| | ModifyDate | C | 8 | | |
| | ModifyTime | C | 8 | | |

**PurchaseDetail**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | * |
| * | PurchaseID | C | 10 | | * |
| * | PurchaseItem | C | 3 | | |
| | ProductID | C | 10 | | * |
| | PuchaseQuatity | C | 12 | | |
| | PurchaseUnitPrice | D | 12 | | |
| | PurchaseAmount | D | 12 | | |

**PurchaseAnalyst**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | PurchaseYear | C | 2 | | |
| * | PurchaseMonth | C | 2 | | |
| * | SupplierID | C | 8 | | * |
| * | ProductID | C | 10 | | * |
| | PuchaseQuatity | D | 12 | 0 | |
| | PurchaseAmount | D | 12 | 0 | |
| | PurchaseReturnQuatity | D | 12 | 0 | |
| | PurchaseReturnAmount | D | 12 | 0 | |

**SupplierInventory**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | SupplierID | C | 8 | | * |
| * | ProductID | C | 10 | | * |
| | Quantity | D | 12 | 0 | |

**SalesAnalyst**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | SalesYear | C | 2 | | |
| * | SalesMonth | C | 2 | | |
| * | CustomerID | C | 8 | | * |
| * | ProductID | C | 10 | | * |
| | SalesQuatity | D | 12 | 0 | |
| | SalesAmount | D | 12 | 0 | |
| | SalesReturnQuatity | D | 12 | 0 | |
| | SalesReturnAmount | D | 12 | 0 | |

**CustomerInventory**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | CustomerID | C | 8 | | * |
| * | ProductID | C | 10 | | * |
| | Quantity | D | 12 | 0 | |

**DeliveryMaster**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | DeliveryID | C | 10 | | |
| | DeliveryDate | C | 8 | | |
| | CustomerID | C | 8 | | * |
| | SalesmanID | C | 4 | | |
| | WarehouseID | C | 4 | | * |
| | DeliveryAddressID | C | 2 | | |
| | InvoiceType | C | 1 | | |
| | InvoiceNo | C | 10 | | |
| | CustomerOrderNo | C | 15 | | |
| | SubTotal | D | 12 | 0 | |
| | ValueAddTax | D | 12 | 0 | |
| | Amount | D | 12 | 0 | |

| | | | | | |
|---|---|---|---|---|---|
| | Sales | D | 12 | 0 | |
| | Tax | D | 12 | 0 | |
| | AccountReceivable | D | 12 | 0 | |
| | Received | D | 12 | 0 | |
| | CreateMan | C | 10 | | |
| | CreateDate | C | 8 | | |
| | CreateTime | C | 8 | | |
| | ModifyMan | C | 10 | | |
| | ModifyDate | C | 8 | | |
| | ModifyTime | C | 8 | | |

**DeliveryDetail**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | * |
| * | DeliveryID | C | 10 | | * |
| * | DeliveryItem | C | 3 | | |
| | ProductID | C | 10 | | * |
| | SalesQuatity | D | 12 | | |
| | SalesUnitPrice | D | 12 | | |
| | SalesAmount | D | 12 | | |
| | SalesAnalystAmount | D | 12 | | |

**InventoryBegining**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | InventoryYear | C | 2 | | |
| * | InventoryMonth | C | 2 | | |
| * | WarehouseID | C | 4 | | * |
| * | ProductID | C | 10 | | * |
| | BeginingQuatity | D | 12 | 0 | |
| | BenginingBalance | D | 12 | 0 | |

**InventoryAnalyst**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | InventoryYear | C | 2 | | |
| * | InventoryMonth | C | 2 | | |
| * | WarehouseID | C | 4 | | * |
| * | ProductID | C | 10 | | * |
| | ChangeCode | C | 2 | | |
| | TotalQuatity | D | 12 | 0 | |
| | TotaAmount | D | 12 | 0 | |

**ChangeMaster**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | ChangeID | C | 10 | | |
| | ChangeDate | C | 8 | | |
| | ChangeCode | C | 2 | | * |
| | WarehouseID | C | 4 | | * |
| | Description | C | 40 | | |
| | CreateMan | C | 10 | | |
| | CreateDate | C | 8 | | |
| | CreateTime | C | 8 | | |
| | ModifyMan | C | 10 | | |
| | ModifyDate | C | 8 | | |
| | ModifyTime | C | 8 | | |

**ChangeDetail**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | * |
| * | ChangeID | C | 10 | | * |
| | ChangeItem | C | 3 | | |
| | ProductID | C | 10 | | |
| | ChangeQuantity | D | 12 | 0 | * |
| | ChangeAmount | D | 12 | 0 | |

**StockTransferMaster**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | TransferID | C | 10 | | |
| | TransferDate | C | 8 | | |
| | TransferOutWarehouseID | C | 4 | | * |
| | TransferInWarehouseID | C | 4 | | * |
| | CreateMan | C | 10 | | |
| | CreateDate | C | 8 | | |
| | CreateTime | C | 8 | | |
| | ModifyMan | C | 10 | | |
| | ModifyDate | C | 8 | | |
| | ModifyTime | C | 8 | | |

**StockTransferDetail**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | * |
| * | TransferID | C | 10 | | * |
| * | TransferItem | C | 3 | | |
| | ProductID | C | 10 | | |
| | TransferQuantity | D | 12 | 0 | * |

**AccountPayableBegining**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | BeginingYear | C | 2 | | |
| * | BeginingMonth | C | 2 | | |
| * | SupplierID | C | 8 | | * |
| | BenginingBalance | D | 12 | 0 | |

**AccountPayableMaster**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | PaymentID | C | 10 | | |
| | PayDate | C | 8 | | |
| | SupplierID | C | 8 | | |
| | PayCash | D | 12 | 0 | |

| | | | | | |
|---|---|---|---|---|---|
| | PayCheque | D | 12 | 0 | * |
| | Discount | D | 12 | 0 | |
| | Remittance | D | 12 | 0 | |
| | Prepaid | D | 12 | 0 | |
| | Others | D | 12 | 0 | |
| | PayAmount | D | 12 | 0 | |
| | TotalBalance | D | 12 | 0 | |
| | CreateMan | C | 10 | | |
| | CreateDate | C | 8 | | |
| | CreateTime | C | 8 | | |
| | ModifyMan | C | 10 | | |
| | ModifyDate | C | 8 | | |
| | ModifyTime | C | 8 | | |

**AccountPayableDetail**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | * |
| * | PaymentID | C | 10 | | * |
| * | PurchaseID | C | 10 | | |
| | Balance | D | 12 | 0 | |

**AccountReceivableBegining**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | BeginingYear | C | 2 | | |
| * | BeginingMonth | C | 2 | | |
| * | CustomerID | C | 8 | | * |
| | BenginingBalance | D | 12 | 0 | |

**AccountReceivableMaster**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | ReceiveID | C | 10 | | |
| | ReceivePayDate | C | 8 | | |
| | CustomerID | C | 8 | | |
| | ReceiveCash | D | 12 | 0 | |
| | ReceivePayCheque | D | 12 | 0 | * |

| | | | | | |
|---|---|---|---|---|---|
| | Discount | D | 12 | 0 | |
| | Remittance | D | 12 | 0 | |
| | Advance | D | 12 | 0 | |
| | Others | D | 12 | 0 | |
| | ReceiveAmount | D | 12 | 0 | |
| | TotalBalance | D | 12 | 0 | |
| | CreateMan | C | 10 | | |
| | CreateDate | C | 8 | | |
| | CreateTime | C | 8 | | |
| | ModifyMan | C | 10 | | |
| | ModifyDate | C | 8 | | |
| | ModifyTime | C | 8 | | |

**AccountReceiveDetail**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | * |
| * | ReceiveID | C | 10 | | * |
| * | SalesID | C | 10 | | |
| | Balance | D | 12 | 0 | |

**Company**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | AttribName | C | 10 | | |
| | CompanyName | C | 60 | | |
| | InvoiceNo | C | 8 | | |
| | OpeningYear | C | 2 | | |
| | OpeningMonth | C | 2 | | |
| | PeriodYear | C | 2 | | |
| | PeriodMonth | C | 2 | | |
| | CreateMan | C | 10 | | |
| | CreateDate | C | 8 | | |
| | CreateTime | C | 8 | | |
| | ModifyMan | C | 10 | | |
| | ModifyDate | C | 8 | | |
| | ModifyTime | C | 8 | | |

**User**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | UserID | C | 10 | | |
| | UserName | C | 10 | | |
| | UserOrGroup | C | 1 | | |
| | GroupID | C | 10 | | |
| | PasswordCode | C | 10 | | |
| | CreateMan | C | 10 | | |
| | CreateDate | C | 8 | | |
| | CreateTime | C | 8 | | |
| | ModifyMan | C | 10 | | |
| | ModifyDate | C | 8 | | |
| | ModifyTime | C | 8 | | |

**Program**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | ProgramID | C | 5 | | |
| | ProgramName | C | 30 | | |
| | ReportOrModify | C | 1 | | |
| | CreateMan | C | 10 | | |
| | CreateDate | C | 8 | | |
| | CreateTime | C | 8 | | |
| | ModifyMan | C | 10 | | |
| | ModifyDate | C | 8 | | |
| | ModifyTime | C | 8 | | |

**UserAuthority**

| Key | Field | Format | Length | Floating | Relation |
|---|---|---|---|---|---|
| * | CompanyID | C | 2 | | |
| * | UserID | C | 10 | | |
| | ProgramID | C | 5 | | |
| | Run | C | 1 | | |
| | Append | C | 1 | | |
| | Edit | C | 1 | | |
| | Del | C | 1 | | |
| | Report | C | 1 | | |
| | BrowseAll | C | 1 | | |
| | EditAll | C | 1 | | |

ote:

C: Common format

D: Digit format

*: Primary / Secondary Key

# SYSTEM IMPLEMENTATION AND TESTING

## 5.0 Introduction

System Implementation involves the translation of the software representation pr0diiccd by the design phase into a computer-readable form. In other words system implication is a process that converts the system requirements and designs into program codes. This phase at times involves some modifications to the previous design. System testing verifies that a system solves the problem as defined by the requirement documents. Sales Purachase & Stock is developed using the top-down approach which involves building the high-level software modules which are further into functions and procedures.

## 5.1 Development Enviroment

The development environment has a very huge impact on the development of a system. The hardware and software chosen to develop the system are very important as they affect the speed of the development progress.

## 5.2 Hardware Used

The hardware used to develop this system are • Pentium 166MI Iz CI[1]U

- 512K Pipeline Burst Cache 32MIl
  RAM
  1.1.1 MI1 Floppy Dtive •
  1.7 GB Hard Disk
- I OX CD-ROM Drive
- WinO)5 1115 Keys Keyboard
- 15" Colour Monitor Canon
  I3.IC-25(>SP

## 5.3 Tools for System Development

The choice of development tools that are chosen plays an important role in determining the lie usability of the system. Besides, development tools must be chosen with the development time in mind. It makes no sense to choose a very sophisticated tool bill not managing to meet the milestones in time. With this in mind, following are the development tools used during the coding phase of Sale, Purchases & Stock System.

| Windows XP | System Requirements | Operating System |
|---|---|---|
| Delphi 6 | System Development | User interface design |
| MS SQL 7 | System Development | Database Design |

## 5.4 PROGRAM CODING

Coding is a process that translates a detail design representation of software into a programming language realization.

### 5.4.1 Methodology

Sales, Purchases & Stock System is developed using a modular approach where each module is developed separately. These modules are then integrated into a fully functional system. For each module, it is further refined into functions and procedures. By applying a modular approach, future modifications and enhancements are made effortless and easy.

### 5.4.2 Coding Principles

The following principles were applied during the implementation of Sales, Purchases & Stock System:

- **Coding conventions**
  Coding conventions such as program labeling, naming conventions, comments and indentation should be adhered to.

- **Readability**
  Codes should be designed adherence to coding readability

- **Maintainability**
  The System should be easily revised or corrected. To facilitate maintenance, code should be readable, modular and as general as possible.

- **Robustness**
  The codes should be able to handle cases of user error by responding appropriately

## 5.5 DATABASE CONNECTIVITY

In order to connect the user section to the database, Open Database Connectivity was created in the server by specifying the Data Source Name (DSN) through the BDE Engine as provide by Delphi programming tool.

## 5.6 TESTING

Testing is a verification and validation process. Verification refers to the set of activities that ensure that the software correctly implements a specific function. On the other hand, validation refers to a different set of activities that ensuring the software has been built traceable to user requirements. Software testing is a critical element of software reliability assurance and represents the ultimate review of requirements specification, design and coding. The objective of testing can be stated as follows:

- Testing is a process of executing a program with the explicit intention of finding error that is making the program fail.

- A good test case is one that has a high probability of finding an as yet undiscovered error.

- A successful lest is one that uncovers an as yet undiscovered error.

Thus, testing is only successful when a fault is discovered or failure occurs as a result of testing procedures. Fault identification is the process of determining what fault or faults caused the failure, and correction or removal is the process of making changes to the system so that the faults are removed.

### 5.6.1 System Testing

System testing is the major quality control measure during prototyping. System testing is to ensure that the entire system is validated. It must be combined with other system element such as hardware, end-user and databases. Testing is performed to ensure that the programs are executed correctly and confirms to the requirements specified. It provides a method to uncover logic error and for testing system reliability. Sales, Purchases & System must achieve accuracy, robustness, flexibility, modularity and reliability.

# SYSTEM EVALUATION

## 6.0 Introduction

During the development process, several problems were encountered in hardware, software, facilities and logic errors in programming the required functions of the system. Some of them were solved at the end, and some will be solved in due time. The system's strengths and limitations were evaluated by variety of users. Improving of the present system and potential enhancements are based on suggestions and evaluations result from these users.

## 6.1 System's Strength

### 6.1.1 User-friendliness

Sales, Purchases & Stock System interfaces are developed through Delphi 6. Thus, the graphical user interface (GUIs) provides users with an easier environment to work on. A standard set of GUI control objects has been applied, such as command buttons, text boxes and list boxes. Especially useful is the list box control because the user need not key in the actual values but need only select them from the given list.

The user-friendly and colorful interface plus the predictable control objects will shorten the learning carve. Consequently, users are more willing to adapt to a new operating system.

### 6.1.2 Searching capabilities

The search engine requires the user to key in the searching criteria before proceeding. This then allows the user to regain their search according their preference, instead of being exposed to a long list of results.

### 6.1.3 Saves time and effort

The system strategy for managing a trading business with various kind of stock item, debtors, creditor is very efficient in the sense that it saves customers' time and effort from manual approaches. They can just by a few click for generating the usable management report.

## 6.4 PROJECT PROBLEMS ENCOUNTERED AND SOLUTIONS

### 6.4.1 Insufficient knowledge in programming technical, databases analysis and meeting various management report

Inadequate comprehension of warehouse trading information and recording system were the major problem encountered during the initial stage. Issues such as daily transactions, administration personal, management regulation and transaction scope in the warehouse trading business had to be thoroughly researched so as to carry out this project. Besides, developing a flawless management system requires extensive studies to understand its requirements. In order to obtain a better understanding, consultancy and interviews with the auditors and accountancies were carried out. Other similar systems were studied.

### 6.4.2 Determining the Project Scope

Due to the time frame given, it was impossible to incorporate too many features into the system. Discussions with the supervisor were held to outline the scope of the system. Availability of tools was also considered in determining the project scope.

## 6.5 SYSTEM LIMITATIONS

### 6.3.1 Caters for general warehouse business only

Sales, Purchases & Stock System only provides general information system management and not other specialized industries and also not incorporated yet with accounting system. This limitations somehow limits the number of user for the system

### 6.3.2 Lack or experience in System programming

As there is no prior knowledge in programming in warehouse environment, a lot of studies need to be performed in order to familiarize with the concept of transaction flow and report requirements programming. On the other hand, programming languages and database development tools need to be learnt within a short span of time.

## CONCLUSION

In conclusion, it can be said that Sales, Purchases & Stock System has met its objective, in developing a warehouse trading business information system. 'The system is easy to learn and use, yet attractive enough to call the attentions of the users.

Developing a warehouse trading application has not been an easy task. However with the proper selection of software used, the development has been made much simpler. Borland Delphi was used to design the user interfaces and the Microsoft SQL 7 was used to create the database. However, due to the time constraint, only minimum requirements have been included. Thus future enhancements have been proposed to improve the functionality of the system.

Much experience has been gained, new knowledge acquired and opportunities available to practice some project management and communication skills.

In view of the above, the objective for a final year project has been achieved which is to give undergraduates an opportunity to undergo different challenges while allowing them the opportunity to experience the whole system development process.

Appendix A

**Customer Information Card**
Standardised ID
Customer name
Person in charges
Contact person
Tel phone No.
Fax No.
Address
Payment method
Term of credit
Category of customer

**Supplier Information Card**
Standardised ID
Supplier name
Person in charges
Contact person
Tel phone No.
Fax No.
Address
Payment method
Category of supplier
Bank current account

**Stock Movement Record**
Movement Date
Record No.
From Wharehouse
To Wharehouse
Item
Product code
Product name
Quantity

**Sales Invoice**
Date of delivery
Delivery Order No.
Wharehouse
Type of sales
- Sales
- Return
- Borrow

Customer name
Standardised ID
Delivery address
Item
Product code
Product name
Quantity
Unit price
Amount
Total
Invoice No.
Sales agent
Note

**Purchases Invoice**
Date of good received
Good Received No.
Wharehouse
Type of sales
- Purchases
- Return
- Borrow

Supplier name
Standardised ID
Delivery address
Item
Product code
Product name
Quantity
Unit price
Amount
Total
Invoice No.
Note

**Stock Card**
Product Name
Product No.
Page
Date
Supporting doc. No.
Move in quantity
Move out quantity
Balance
Note

**Collection list**
Date
Official receipt no.
Customer Name
Cash
Cheque
- Cheque No.
- Due date
- Amount
- Bank
- Total
- No. of cheque

Sales Invoice No.
Unpaid balance
Knock off amount
Currrent receipt
Account
Representative

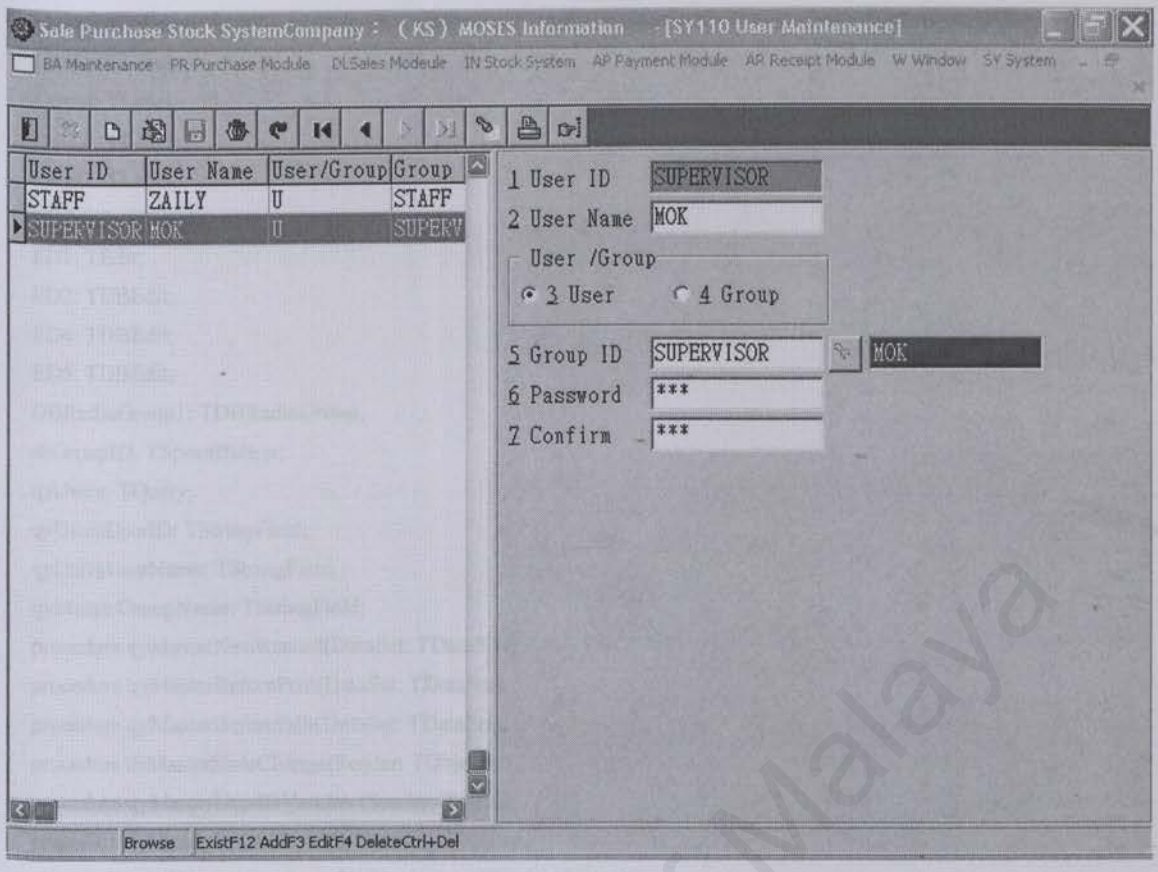
**Payment list**
Date
Official receipt no.
Supplier name
Cash
Cheque
- Cheque No.
- Due date
- Amount
- Bank
- Total
- No. of cheque

Purchase Invoice No.
Unpaid balance
Knock off amount
Currrent Payment
Account
Representative

**Balance of Stock Record**

Movement Date
Record No.
Nature of movement
- To
- From

Reason of movement
Item
Product code
Product name
Quantity



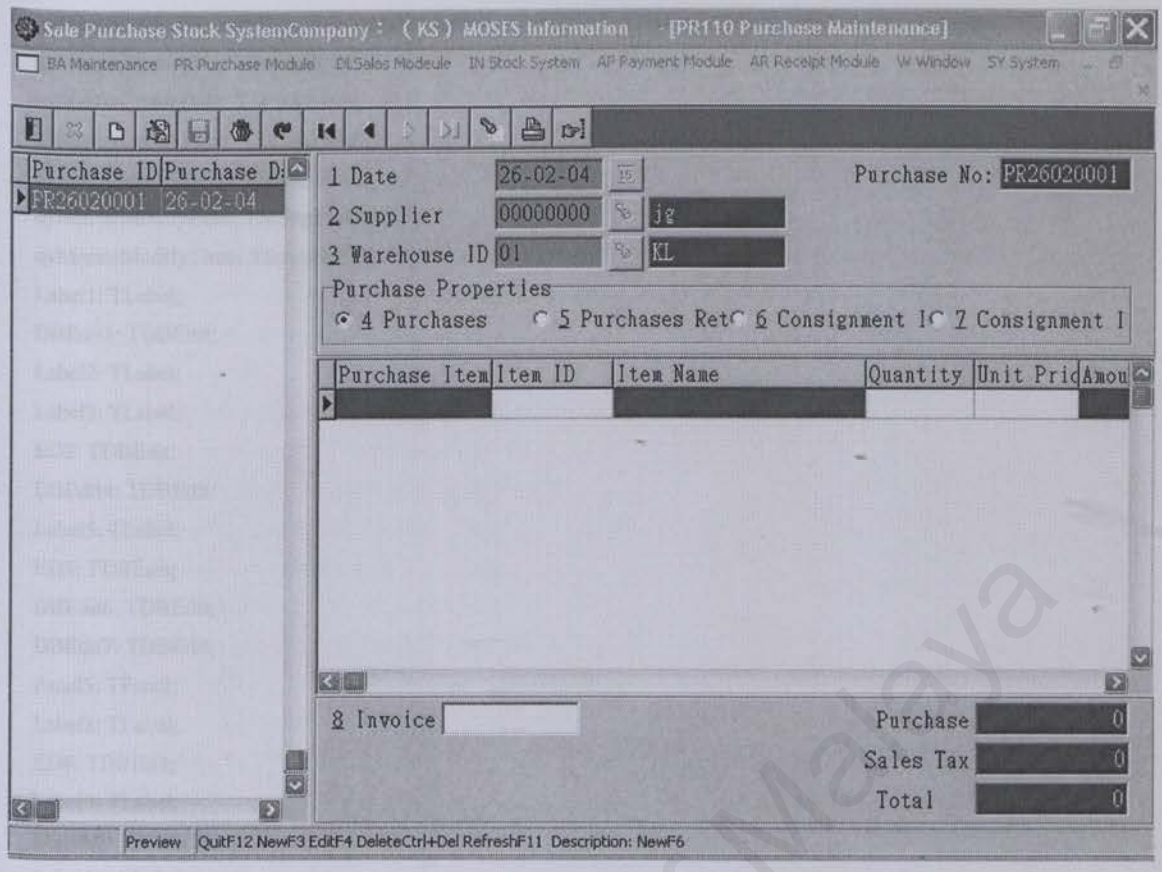

```
unit SY110;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Single, Menus, Db, DBTables, Grids, DBGrids, Buttons, ExtCtrls, StdCtrls,
  DBCtrls, Mask;

type
  TfmSY110 = class(TfmSingle)
    qyMasterCompanyID: TStringField;
    qyMasterUserID: TStringField;
    qyMasterUserName: TStringField;
    qyMasterUserOrGroup: TStringField;
    qyMasterGroupID: TStringField;
    qyMasterPasswordCode: TStringField;
    qyMasterCreateMan: TStringField;
    qyMasterCreateDate: TStringField;
    qyMasterCreateTime: TStringField;
    qyMasterModifyMan: TStringField;
    qyMasterModifyDate: TStringField;
    qyMasterModifyTime: TStringField;
```

```
    Label1: TLabel;
    Label2: TLabel;
    Label4: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    DBEdit2: TDBEdit;
    ED7: TEdit;
    ED2: TDBEdit;
    ED6: TDBEdit;
    ED5: TDBEdit;
    DBRadioGroup1: TDBRadioGroup;
    sbGroupID: TSpeedButton;
    qyUsers: TQuery;
    qyUsersUserID: TStringField;
    qyUsersUserName: TStringField;
    qyMasterGroupName: TStringField;
    procedure qyMasterNewRecord(DataSet: TDataSet);
    procedure qyMasterBeforePost(DataSet: TDataSet);
    procedure qyMasterBeforeEdit(DataSet: TDataSet);
    procedure dsMasterStateChange(Sender: TObject);
    procedure qyMasterUserIDValidate(Sender: TField);
    procedure qyMasterUpdateRecord(DataSet: TDataSet;
      UpdateKind: TUpdateKind; var UpdateAction: TUpdateAction);
    procedure sbGroupIDClick(Sender: TObject);
    procedure qyMasterGroupIDValidate(Sender: TField);
  private
    { Private declarations }
  public
    { Public declarations }

    procedure OpenDB; override;
  end;

var
  fmSY110: TfmSY110;

implementation

uses Main, DataModule, PublicFunction, CheckData, BaseSearch, GetData;

{$R *.DFM}

{ TfmSY110 }

procedure TfmSY110.OpenDB;
begin
```

```
 qyMaster.Close;
 OrderBySQL := 'M.UserID';
 qyMaster.SQL.Text := SQLText;
 qyMaster.Open;
end;

procedure TfmSY110.qyMasterNewRecord(DataSet: TDataSet);
begin
 inherited;

 with qyMaster do
 begin
  FieldByName('UserID').AsString := Space(10);
  FieldByName('UserName').AsString := Space(8);
  FieldByName('UserOrGroup').AsString := 'U';
  FieldByName('GroupID').AsString := Space(10);
  FieldByName('PasswordCode').AsString := Space(10);
 end;
 ED7.Text := '';
end;

procedure TfmSY110.qyMasterBeforePost(DataSet: TDataSet);
begin
 inherited;

 if qyMaster.FieldByName('UserOrGroup').AsString = 'G' then
  qyMaster.FieldByName('GroupID').AsString := qyMaster.FieldByName('UserID').AsString;
 if qyMaster.State = dsInsert then
 begin
  if Trim(qyMaster.FieldByName('UserID').AsString) = '' then
  begin
   NullWarning('User ID');
   ED1.SetFocus;
   Abort;
  end;
  with qyTemp do
  begin
   Close;
   SQL.Clear;
   SQL.Add('SELECT UserID ');
   SQL.Add('FROM Users ');
   SQL.Add('WHERE CompanyID = :CompanyID AND UserID = :UserID ');
   ParamByName('CompanyID').AsString := sCompanyID;
   ParamByName('UserID').AsString := qyMaster.FieldByName('UserID').AsString;
   Open;
  end;
```

```
  if qyTemp.FieldByName('UserID').AsString <> '' then
  begin
   RepeatWarning('User ID');
   ED1.SetFocus;
   Abort;
  end;
 end;
 if Trim(qyMaster.FieldByName('UserName').AsString) = '' then
 begin
  NullWarning('User ID');
  ED2.SetFocus;
  Abort;
 end;
 if (qyMaster.FieldByName('UserOrGroup').AsString = 'U') and
   (qyMaster.FieldByName('UserID').AsString <> qyMaster.FieldByName('GroupID').AsString) then
 begin
  if not CheckUserID(qyMaster.FieldByName('GroupID').AsString) then
  begin
   ED5.SetFocus;
   Abort;
  end;
 end;
 if Trim(qyMaster.FieldByName('PasswordCode').AsString) = '' then
 begin
  NullWarning('Password');
  ED6.SetFocus;
  Abort;
 end;
 if UpperCase(ED6.Text) <> UpperCase(ED7.Text) then
 begin
  MyWarning('Password must be consistence to Confirmation Password    ');
  ED6.SetFocus;
  Abort;
 end;
 qyMaster.FieldByName('PasswordCode').AsString := UpperCase(ED6.Text);
end;

procedure TfmSY110.qyMasterBeforeEdit(DataSet: TDataSet);
begin
 inherited;
 ED7.Text := qyMaster.FieldByName('PasswordCode').AsString;
end;

procedure TfmSY110.dsMasterStateChange(Sender: TObject);
begin
 inherited;
```

```
 sbGroupID.Enabled := (qyMaster.State in [dsInsert, dsEdit]);
 ED7.ReadOnly := (not (qyMaster.State in [dsInsert, dsEdit]));
end;

procedure TfmSY110.qyMasterUserIDValidate(Sender: TField);
begin
 inherited;
 if Trim(qyMaster.FieldByName('GroupID').AsString) = '' then
  qyMaster.FieldByName('GroupID').AsString := qyMaster.FieldByName('UserID').AsString;
end;

procedure TfmSY110.qyMasterUpdateRecord(DataSet: TDataSet;
 UpdateKind: TUpdateKind; var UpdateAction: TUpdateAction);
begin
 if UpdateKind = ukDelete then
 begin
  with qyTemp do
  begin
   Close;
   SQL.Clear;
   SQL.Add('DELETE FROM UserAuthority ');
   SQL.Add('WHERE CompanyID = :CompanyID AND UserID = :UserID ');
   ParamByName('CompanyID').AsString := sCompanyID;
   ParamByName('UserID').AsString := DataSet.FieldByName('UserID').OldValue;
   ExecSQL;
  end;
 end;
 inherited;
end;

procedure TfmSY110.sbGroupIDClick(Sender: TObject);
begin
 inherited;
 SearchData(ED5, qyUsers);
end;

procedure TfmSY110.qyMasterGroupIDValidate(Sender: TField);
begin
 inherited;
 qyMaster.FieldByName('GroupName').AsString :=
      GetUsersName(qyMaster.FieldByName('GroupID').AsString);
end;

end.
```




```
unit PR110;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  MasterDetail, Menus, Db, DBTables, Grids, DBGrids, Buttons, ExtCtrls,
  StdCtrls, Mask, DBCtrls;

type
  TfmPR110 = class(TfmMasterDetail)
    qyMasterSupplierAttribName: TStringField;
    qyMasterWarehouseName: TStringField;
    qyMasterCompanyID: TStringField;
    qyMasterWarehouseID: TStringField;
    qyMasterPurchaseID: TStringField;
    qyMasterPurchaseDate: TStringField;
    qyMasterSupplierID: TStringField;
    qyMasterPurchaseProperty: TStringField;
    qyMasterInvoiceNo: TStringField;
    qyMasterSubTotal: TBCDField;
    qyMasterValueAddTax: TBCDField;
    qyMasterAmount: TBCDField;
```

```
qyMasterPaid: TBCDField;
qyMasterCreateMan: TStringField;
qyMasterCreateDate: TStringField;
qyMasterCreateTime: TStringField;
qyMasterModifyMan: TStringField;
qyMasterModifyDate: TStringField;
qyMasterModifyTime: TStringField;
Label1: TLabel;
DBEdit1: TDBEdit;
Label2: TLabel;
Label3: TLabel;
ED2: TDBEdit;
DBEdit4: TDBEdit;
Label5: TLabel;
ED3: TDBEdit;
DBEdit6: TDBEdit;
DBEdit7: TDBEdit;
Panel5: TPanel;
Label8: TLabel;
ED8: TDBEdit;
Label9: TLabel;
DBEdit9: TDBEdit;
Label10: TLabel;
DBEdit10: TDBEdit;
Label11: TLabel;
DBEdit11: TDBEdit;
sbSupplierID: TSpeedButton;
sbPurchaseDate: TSpeedButton;
sbWarehouseID: TSpeedButton;
rgPurchaseProperty: TDBRadioGroup;
qyDetailProductName: TStringField;
qyDetailCompanyID: TStringField;
qyDetailPurchaseID: TStringField;
qyDetailPurchaseItem: TStringField;
qyDetailProductID: TStringField;
qyDetailPurchaseQuantity: TBCDField;
qyDetailPurchaseUnitPrice: TBCDField;
qyDetailPurchaseAmount: TBCDField;
qyTemp2: TQuery;
qyTemp1: TQuery;
edProductID: TDBEdit;
qyMasterAccountPayable: TBCDField;
procedure FormCreate(Sender: TObject);
procedure qyMasterBeforeOpen(DataSet: TDataSet);
procedure qyMasterNewRecord(DataSet: TDataSet);
procedure qyMasterBeforePost(DataSet: TDataSet);
```

```
    procedure qyDetailBeforeOpen(DataSet: TDataSet);
    procedure qyDetailNewRecord(DataSet: TDataSet);
    procedure qyDetailBeforePost(DataSet: TDataSet);
    procedure sbPurchaseDateClick(Sender: TObject);
    procedure sbSupplierIDClick(Sender: TObject);
    procedure sbWarehouseIDClick(Sender: TObject);
    procedure qyDetailPurchaseQuantityValidate(Sender: TField);
    procedure qyDetailPurchaseUnitPriceValidate(Sender: TField);
    procedure qyDetailUpdateRecord(DataSet: TDataSet;
      UpdateKind: TUpdateKind; var UpdateAction: TUpdateAction);
    procedure qyMasterSupplierIDValidate(Sender: TField);
    procedure qyMasterWarehouseIDValidate(Sender: TField);
    procedure qyDetailProductIDValidate(Sender: TField);
    procedure gdDetailEditButtonClick(Sender: TObject);
    procedure qyDetailAfterInsert(DataSet: TDataSet);
    procedure dsMasterStateChange(Sender: TObject);
    procedure qyMasterUpdateRecord(DataSet: TDataSet;
      UpdateKind: TUpdateKind; var UpdateAction: TUpdateAction);
    procedure sbSelectClick(Sender: TObject);
    procedure sbReportClick(Sender: TObject);
  private
    { Private declarations }
    procedure CalcPurchaseAmount;
    procedure UpdatePurchaseAnalyst(PurchaseDate, SupplierID, ProductID,
        PurchaseProperty: String; PurchaseQuantity, PurchaseAmount: Extended);
    procedure UpdateSupplierInventory(SupplierID, ProductID: String;
        Quantity: Extended);
  public
    { Public declarations }
    AStartPeriodDate, AEndPeriodDate : String;
    AStartSupplierID, AEndSupplierID : String;
    AStartWarehouseID, AEndWarehouseID : String;
    AProperty5, AProperty6, AProperty7, AProperty8 : String;
    ReData : Boolean;
    procedure OpenDB; override;
    procedure CalcAmount; override;
    procedure GetPrevValue; override;
  end;

var
  fmPR110: TfmPR110;

implementation

uses Main, DataModule, PublicFunction, CheckData, BaseSearch,
  GetData, PR110Select, Loading, PR110Report, CalendarSearch;
```

```
var
  sPrevPurchaseDate, sPrevSupplierID, sPrevWarehouseID, sPrevPurchaseProperty : String;

{$R *.DFM}

{ TfmPR110 }

procedure TfmPR110.FormCreate(Sender: TObject);
begin
  AStartPeriodDate := sStartPeriodDate;
  AEndPeriodDate := sEndPeriodDate;
  AStartSupplierID := '00000000';
  AEndSupplierID := 'ZZZZZZZZ';
  AStartWarehouseID := '00';
  AEndWarehouseID := 'ZZ';
  AProperty5 := '5';
  AProperty6 := '6';
  AProperty7 := '7';
  AProperty8 := '8';
  ReData := True;
  inherited;
  sPrevPurchaseDate := '';
  sPrevSupplierID := '';
  sPrevWarehouseID := '';
  sPrevPurchaseProperty := '';
end;

procedure TfmPR110.OpenDB;
begin
  qyMaster.Close;
  OrderBySQL := 'M.PurchaseID';
  qyMaster.SQL.Text := SQLText;
  qyMaster.Open;
  ItemFieldName := 'PurchaseItem';
  DataDate := 'PurchaseDate';
end;

procedure TfmPR110.CalcAmount;
var
  fSubTotal : Extended;
  PrevBookmark : TBookmark;
begin

  fSubTotal := 0;
```

```
  PrevBookmark := qyDetail.GetBookmark;
  try
   qyDetail.DisableControls;
   qyDetail.First;
   while not qyDetail.Eof do
   begin
    fSubTotal := fSubTotal + qyDetail.FieldByName('PurchaseAmount').AsFloat;
    qyDetail.Next;
   end;
   qyMaster.FieldByName('SubTotal').AsFloat := fSubTotal;
   qyMaster.FieldByName('ValueAddTax').AsFloat :=
       RealToInt(qyMaster.FieldByName('SubTotal').AsFloat * 0.05);
   qyMaster.FieldByName('Amount').AsFloat :=
       qyMaster.FieldByName('SubTotal').AsFloat +
       qyMaster.FieldByName('ValueAddTax').AsFloat;
  finally
   qyDetail.EnableControls;
   if PrevBookmark <> nil then
   begin
    qyDetail.GotoBookmark(PrevBookmark);
    qyDetail.FreeBookmark(PrevBookmark);
   end;
  end;
end;

procedure TfmPR110.qyMasterBeforeOpen(DataSet: TDataSet);
begin
 inherited;

 with qyMaster do
 begin
  ParamByName('StartPeriodDate').AsString := AStartPeriodDate;
  ParamByName('EndPeriodDate').AsString := AEndPeriodDate;
  ParamByName('StartSupplierID').AsString := AStartSupplierID;
  ParamByName('EndSupplierID').AsString := AEndSupplierID;
  ParamByName('StartWarehouseID').AsString := AStartWarehouseID;
  ParamByName('EndWarehouseID').AsString := AEndWarehouseID;
  ParamByName('Property1').AsString := AProperty5;
  ParamByName('Property2').AsString := AProperty6;
  ParamByName('Property3').AsString := AProperty7;
  ParamByName('Property4').AsString := AProperty8;
 end;
end;

procedure TfmPR110.qyMasterNewRecord(DataSet: TDataSet);
begin
```

```
 inherited;

 with qyMaster do
 begin
  FieldByName('PurchaseID').AsString := Space(10);
  FieldByName('PurchaseDate').AsString := Today;
  FieldByName('SupplierID').AsString := Space(8);
  FieldByName('WarehouseID').AsString := Space(2);
  FieldByName('PurchaseProperty').AsString := '5';
  FieldByName('InvoiceNo').AsString := Space(8);
  FieldByName('SubTotal').AsFloat := 0;
  FieldByName('ValueAddTax').AsFloat := 0;
  FieldByName('Amount').AsFloat := 0;
  FieldByName('AccountPayable').AsFloat := 0;
  FieldByName('Paid').AsFloat := 0;
 end;
end;

procedure TfmPR110.qyMasterBeforePost(DataSet: TDataSet);
begin
 inherited;

 if qyMaster.State = dsInsert then
 begin
  if not CheckDate(qyMaster.FieldByName('PurchaseDate').AsString) then
  begin
   ED1.SetFocus;
   Abort;
  end;
  if not CheckSupplierID(qyMaster.FieldByName('SupplierID').AsString) then
  begin
   ED2.SetFocus;
   Abort;
  end;
  if not CheckWarehouseID(qyMaster.FieldByName('WarehouseID').AsString) then
  begin
   ED3.SetFocus;
   Abort;
  end;
  qyMaster.FieldByName('PurchaseID').AsString :=
       IDGen('PR', qyMaster.FieldByName('PurchaseDate').AsString,
           'PurchaseID', 'PurchaseMaster');
 end;
 if (qyMaster.FieldByName('PurchaseProperty').AsString = '5') then
  qyMaster.FieldByName('AccountPayable').AsFloat :=
       qyMaster.FieldByName('Amount').AsFloat
```

```
  else
   if (qyMaster.FieldByName('PurchaseProperty').AsString = '6') then
    qyMaster.FieldByName('AccountPayable').AsFloat :=
        qyMaster.FieldByName('Amount').AsFloat * -1
   else
    qyMaster.FieldByName('AccountPayable').AsFloat := 0;
end;

procedure TfmPR110.qyDetailBeforeOpen(DataSet: TDataSet);
begin
 inherited;

 qyDetail.ParamByName('PurchaseID').AsString := qyMaster.FieldByName('PurchaseID').AsString;
end;

procedure TfmPR110.qyDetailNewRecord(DataSet: TDataSet);
begin
 inherited;

 with qyDetail do
 begin
  FieldByName('PurchaseID').AsString := qyMaster.FieldByName('PurchaseID').AsString;
  FieldByName('PurchaseItem').AsString := Space(3);
  FieldByName('ProductID').AsString := Space(10);
  FieldByName('PurchaseQuantity').AsFloat := 0;
  FieldByName('PurchaseUnitPrice').AsFloat := 0;
  FieldByName('PurchaseAmount').AsFloat := 0;
 end;
end;

procedure TfmPR110.qyDetailBeforePost(DataSet: TDataSet);
begin
 inherited;

 if not CheckProductID(qyDetail.FieldByName('ProductID').AsString) then
 begin
  gdDetail.SelectedField := qyDetailProductID;
  Abort;
 end;
 CalcPurchaseAmount;
 if qyDetail.State = dsInsert then
  qyDetail.FieldByName('PurchaseItem').AsString :=
      ITGen(qyMaster.FieldByName('PurchaseID').AsString,
        'PurchaseItem', 'PurchaseID', 'PurchaseDetail');
end;
```

```
procedure TfmPR110.sbPurchaseDateClick(Sender: TObject);
begin
  inherited;
// MyCalendar(ED1);
  ShowCalendar(ED1);
end;

procedure TfmPR110.sbSupplierIDClick(Sender: TObject);
begin
  inherited;
  qyMaster.FieldByName('SupplierID').AsString := SearchData(ED2, DM.qySupplier);
end;

procedure TfmPR110.sbWarehouseIDClick(Sender: TObject);
begin
  inherited;
  qyMaster.FieldByName('WarehouseID').AsString := SearchData(ED3, DM.qyWarehouse);
end;

procedure TfmPR110.CalcPurchaseAmount;
begin
  qyDetail.FieldByName('PurchaseAmount').AsFloat :=
        qyDetail.FieldByName('PurchaseQuantity').AsFloat *
        qyDetail.FieldByName('PurchaseUnitPrice').AsFloat;
end;

procedure TfmPR110.qyDetailPurchaseQuantityValidate(Sender: TField);
begin
  inherited;
  CalcPurchaseAmount;
end;

procedure TfmPR110.qyDetailPurchaseUnitPriceValidate(Sender: TField);
begin
  inherited;
  CalcPurchaseAmount;
end;

procedure TfmPR110.qyDetailUpdateRecord(DataSet: TDataSet;
  UpdateKind: TUpdateKind; var UpdateAction: TUpdateAction);
var
  fNewPurchaseQuantity, fOldPurchaseQuantity : Extended;
  fNewPurchaseAmount, fOldPurchaseAmount : Extended;
  sNewProductID, sOldProductID : String;
begin
  inherited;
```

```
if ReSortting then Exit;
fNewPurchaseQuantity := 0;
fOldPurchaseQuantity := 0;
fNewPurchaseAmount := 0;
fOldPurchaseAmount := 0;
sNewProductID := '';
sOldProductID := '';
case UpdateKind of
 ukInsert :
  begin
   fNewPurchaseQuantity := DataSet.FieldByName('PurchaseQuantity').NewValue;
   fOldPurchaseQuantity := 0;
   fNewPurchaseAmount := DataSet.FieldByName('PurchaseAmount').NewValue;
   fOldPurchaseAmount := 0;
   sNewProductID := DataSet.FieldByName('ProductID').NewValue;
   sOldProductID := DataSet.FieldByName('ProductID').NewValue;
   with DM.qyTemp1 do
   begin
    Close;
    SQL.Clear;
    SQL.Add('UPDATE Product SET LastPurchaseDate = :LastPurchaseDate ');
    SQL.Add('WHERE CompanyID = :CompanyID AND ProductID = :ProductID ');
    ParamByName('CompanyID').AsString := sCompanyID;
    ParamByName('ProductID').AsString := DataSet.FieldByName('ProductID').AsString;
    ParamByName('LastPurchaseDate').AsString := qyMaster.FieldByName('PurchaseDate').AsString;
    ExecSQL;
   end;
  end;
 ukModify :
  begin
   fNewPurchaseQuantity := DataSet.FieldByName('PurchaseQuantity').NewValue;
   fOldPurchaseQuantity := DataSet.FieldByName('PurchaseQuantity').OldValue;
   fNewPurchaseAmount := DataSet.FieldByName('PurchaseAmount').NewValue;
   fOldPurchaseAmount := DataSet.FieldByName('PurchaseAmount').OldValue;
   sNewProductID := DataSet.FieldByName('ProductID').NewValue;
   sOldProductID := DataSet.FieldByName('ProductID').OldValue;
  end;
 ukDelete :
  begin
   fNewPurchaseQuantity := 0;
   fOldPurchaseQuantity := DataSet.FieldByName('PurchaseQuantity').OldValue;
   fNewPurchaseAmount := 0;
   fOldPurchaseAmount := DataSet.FieldByName('PurchaseAmount').OldValue;
   sNewProductID := DataSet.FieldByName('ProductID').OldValue;
   sOldProductID := DataSet.FieldByName('ProductID').OldValue;
  end;
```

```
  end;
  if (sPrevPurchaseProperty = '5') or (sPrevPurchaseProperty = '6') then
  begin
   UpdatePurchaseAnalyst(sPrevPurchaseDate, sPrevSupplierID,
                sOldProductID, sPrevPurchaseProperty,
                (fOldPurchaseQuantity * -1),
                (fOldPurchaseAmount * -1));
   UpdatePurchaseAnalyst(sPrevPurchaseDate, sPrevSupplierID,
                sNewProductID, sPrevPurchaseProperty,
                (fNewPurchaseQuantity),
                (fNewPurchaseAmount));
  end;
  if (sPrevPurchaseProperty = '7') then
  begin
   UpdateSupplierInventory(sPrevSupplierID, sOldProductID,
                (fOldPurchaseQuantity * -1));
   UpdateSupplierInventory(sPrevSupplierID, sNewProductID,
                fNewPurchaseQuantity);
  end;
  if (sPrevPurchaseProperty = '8') then
  begin
   UpdateSupplierInventory(sPrevSupplierID, sOldProductID,
                (fOldPurchaseQuantity));
   UpdateSupplierInventory(sPrevSupplierID, sNewProductID,
                fNewPurchaseQuantity * -1);
  end;
  UpdateInventoryAnalyst(sPrevPurchaseDate, sOldProductID,
               sPrevWarehouseID, '0' + sPrevPurchaseProperty,
               (fOldPurchaseQuantity * -1),
               (fOldPurchaseAmount * -1));
  UpdateInventoryAnalyst(sPrevPurchaseDate, sNewProductID,
               sPrevWarehouseID, '0' + sPrevPurchaseProperty,
               fNewPurchaseQuantity,
               fNewPurchaseAmount);
end;

procedure TfmPR110.UpdatePurchaseAnalyst(PurchaseDate, SupplierID,
  ProductID, PurchaseProperty: String; PurchaseQuantity,
  PurchaseAmount: Extended);
begin
  with qyTemp do
  begin
   Close;
   SQL.Clear;
   SQL.Add('SELECT SupplierID ');
   SQL.Add('FROM PurchaseAnalyst ');
```

```
  SQL.Add('WHERE CompanyID = :CompanyID ');
  SQL.Add('AND PurchaseYear = :PurchaseYear AND PurchaseMonth = :PurchaseMonth ');
  SQL.Add('AND SupplierID = :SupplierID AND ProductID = :ProductID ');
  ParamByName('CompanyID').AsString := sCompanyID;
  ParamByName('PurchaseYear').AsString := RightStr(PurchaseDate, 2);
  ParamByName('PurchaseMonth').AsString := SubStr(PurchaseDate, 4, 2);
  ParamByName('SupplierID').AsString := SupplierID;
  ParamByName('ProductID').AsString := ProductID;
  Open;
 end;
 if qyTemp.FieldByName('SupplierID').AsString = '' then
 begin
  with qyTemp2 do
  begin
   Close;
   SQL.Clear;
   SQL.Add('INSERT INTO PurchaseAnalyst ');
   SQL.Add(' (CompanyID, PurchaseYear, PurchaseMonth, SupplierID, ProductID, ');
   SQL.Add('  PurchaseQuantity, PurchaseAmount, PurchaseReturnQuantity, PurchaseReturnAmount) ');
   SQL.Add('VALUES ');
   SQL.Add(' (:CompanyID, :PurchaseYear, :PurchaseMonth, :SupplierID, :ProductID, ');
   SQL.Add('  :PurchaseQuantity, :PurchaseAmount, :PurchaseReturnQuantity, :PurchaseReturnAmount) ');
   ParamByName('CompanyID').AsString := sCompanyID;
   ParamByName('PurchaseYear').AsString := RightStr(PurchaseDate, 2);
   ParamByName('PurchaseMonth').AsString := SubStr(PurchaseDate, 4, 2);
   ParamByName('SupplierID').AsString := SupplierID;
   ParamByName('ProductID').AsString := ProductID;
   if PurchaseProperty = '5' then
   begin
    ParamByName('PurchaseQuantity').AsFloat := PurchaseQuantity;
    ParamByName('PurchaseAmount').AsFloat := PurchaseAmount;
    ParamByName('PurchaseReturnQuantity').AsFloat := 0;
    ParamByName('PurchaseReturnAmount').AsFloat := 0;
   end
   else
   begin
    ParamByName('PurchaseQuantity').AsFloat := 0;
    ParamByName('PurchaseAmount').AsFloat := 0;
    ParamByName('PurchaseReturnQuantity').AsFloat := PurchaseQuantity;
    ParamByName('PurchaseReturnAmount').AsFloat := PurchaseAmount;
   end;
   ExecSQL;
  end;
 end
 else
 begin
```

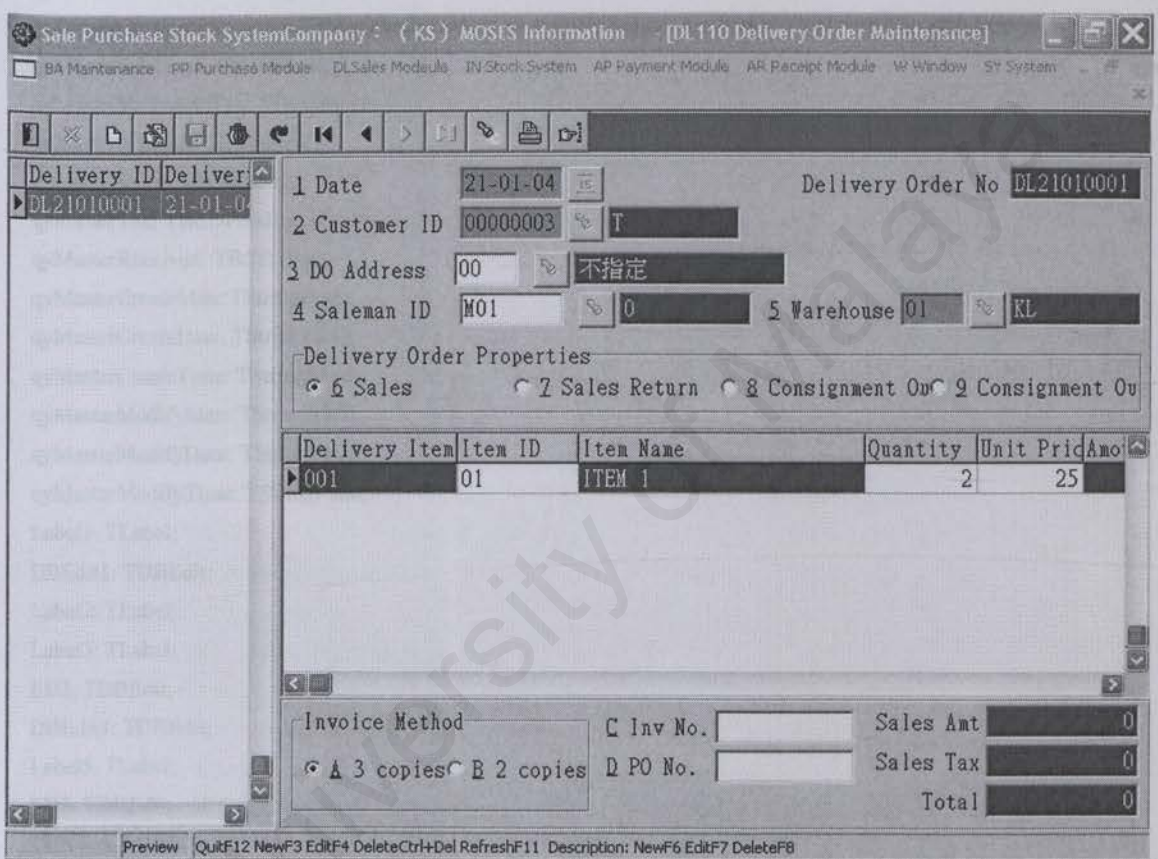
```
    with qyTemp2 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('UPDATE PurchaseAnalyst ');
      SQL.Add('SET PurchaseQuantity = PurchaseQuantity + :PurchaseQuantity, ');
      SQL.Add('    PurchaseAmount = PurchaseAmount + :PurchaseAmount, ');
      SQL.Add('    PurchaseReturnQuantity = PurchaseReturnQuantity + :PurchaseReturnQuantity, ');
      SQL.Add('    PurchaseReturnAmount = PurchaseReturnAmount + :PurchaseReturnAmount ');
      SQL.Add('WHERE CompanyID = :CompanyID ');
      SQL.Add('AND PurchaseYear = :PurchaseYear AND PurchaseMonth = :PurchaseMonth ');
      SQL.Add('AND SupplierID = :SupplierID AND ProductID = :ProductID ');
      ParamByName('CompanyID').AsString := sCompanyID;
      ParamByName('PurchaseYear').AsString := RightStr(PurchaseDate, 2);
      ParamByName('PurchaseMonth').AsString := SubStr(PurchaseDate, 4, 2);
      ParamByName('SupplierID').AsString := SupplierID;
      ParamByName('ProductID').AsString := ProductID;
      if PurchaseProperty = '5' then
      begin
        ParamByName('PurchaseQuantity').AsFloat := PurchaseQuantity;
        ParamByName('PurchaseAmount').AsFloat := PurchaseAmount;
        ParamByName('PurchaseReturnQuantity').AsFloat := 0;
        ParamByName('PurchaseReturnAmount').AsFloat := 0;
      end
      else
      begin
        ParamByName('PurchaseQuantity').AsFloat := 0;
        ParamByName('PurchaseAmount').AsFloat := 0;
        ParamByName('PurchaseReturnQuantity').AsFloat := PurchaseQuantity;
        ParamByName('PurchaseReturnAmount').AsFloat := PurchaseAmount;
      end;
      ExecSQL;
    end;
  end;
end;

procedure TfmPR110.UpdateSupplierInventory(SupplierID, ProductID: String;
  Quantity: Extended);
begin
  with qyTemp do
  begin
    Close;
    SQL.Clear;
    SQL.Add('SELECT SupplierID ');
    SQL.Add('FROM SupplierInventory ');
    SQL.Add('WHERE CompanyID = :CompanyID ');
```

```
    SQL.Add('AND SupplierID = :SupplierID AND ProductID = :ProductID ');
    ParamByName('CompanyID').AsString := sCompanyID;
    ParamByName('SupplierID').AsString := SupplierID;
    ParamByName('ProductID').AsString := ProductID;
    Open;
   end;
   if qyTemp.FieldByName('SupplierID').AsString = '' then
   begin
    with qyTemp2 do
    begin
     Close;
     SQL.Clear;
     SQL.Add('INSERT INTO SupplierInventory ');
     SQL.Add(' (CompanyID, SupplierID, ProductID, Quantity)');
     SQL.Add('VALUES ');
     SQL.Add(' (:CompanyID, :SupplierID, :ProductID, :Quantity) ');
     ParamByName('CompanyID').AsString := sCompanyID;
     ParamByName('SupplierID').AsString := SupplierID;
     ParamByName('ProductID').AsString := ProductID;
     ParamByName('Quantity').AsFloat := Quantity;
     ExecSQL;
    end;
   end
   else
   begin
    with qyTemp2 do
    begin
     Close;
     SQL.Clear;
     SQL.Add('UPDATE SupplierInventory ');
     SQL.Add('SET Quantity = Quantity + :Quantity ');
     SQL.Add('WHERE CompanyID = :CompanyID ');
     SQL.Add('AND SupplierID = :SupplierID AND ProductID = :ProductID ');
     ParamByName('CompanyID').AsString := sCompanyID;
     ParamByName('SupplierID').AsString := SupplierID;
     ParamByName('ProductID').AsString := ProductID;
     ParamByName('Quantity').AsFloat := Quantity;
     ExecSQL;
    end;
   end;
  end;

  procedure TfmPR110.qyMasterSupplierIDValidate(Sender: TField);
  begin
   inherited;
   qyMaster.FieldByName('SupplierAttribName').AsString :=
```

```
    GetSupplierAttribName(qyMaster.FieldByName('SupplierID').AsString);
end;

procedure TfmPR110.qyMasterWarehouseIDValidate(Sender: TField);
begin
 inherited;
 qyMaster.FieldByName('WarehouseName').AsString :=
    GetWarehouseName(qyMaster.FieldByName('WarehouseID').AsString);
end;

procedure TfmPR110.qyDetailProductIDValidate(Sender: TField);
begin
 inherited;
 qyDetail.FieldByName('ProductName').AsString :=
    GetProductName(qyDetail.FieldByName('ProductID').AsString);
end;

procedure TfmPR110.gdDetailEditButtonClick(Sender: TObject);
begin
 inherited;
 if (gdDetail.SelectedField.FieldName = 'ProductID') and
   (qyDetail.State in [dsInsert, dsEdit]) then
  qyDetail.FieldByName('ProductID').AsString := SearchData(edProductID, DM.qyProduct);
end;

procedure TfmPR110.qyDetailAfterInsert(DataSet: TDataSet);
begin
 inherited;
 gdDetail.SelectedField := qyDetailProductID;
end;

procedure TfmPR110.dsMasterStateChange(Sender: TObject);
begin
 inherited;
 rgPurchaseProperty.ReadOnly := (qyMaster.State <> dsInsert);
 ED2.ReadOnly := (qyMaster.State <> dsInsert);
 ED3.ReadOnly := (qyMaster.State <> dsInsert);
 sbPurchaseDate.Enabled := (qyMaster.State = dsInsert);
 sbSupplierID.Enabled := (qyMaster.State = dsInsert);
 sbWarehouseID.Enabled := (qyMaster.State = dsInsert);
end;

procedure TfmPR110.GetPrevValue;
begin
 inherited;
 sPrevPurchaseDate := qyMaster.FieldByName('PurchaseDate').AsString;
```

```
  sPrevSupplierID := qyMaster.FieldByName('SupplierID').AsString;
  sPrevWarehouseID := qyMaster.FieldByName('WarehouseID').AsString;
  sPrevPurchaseProperty := qyMaster.FieldByName('PurchaseProperty').AsString;
end;

procedure TfmPR110.qyMasterUpdateRecord(DataSet: TDataSet;
  UpdateKind: TUpdateKind; var UpdateAction: TUpdateAction);
begin
  inherited;
  if UpdateKind = ukInsert then
  begin
    with qyTemp do
    begin
      Close;
      SQL.Clear;
      SQL.Add('UPDATE Supplier SET LastPurchaseDate = :LastPurchaseDate ');
      SQL.Add('WHERE CompanyID = :CompanyID AND SupplierID = :SupplierID ');
      ParamByName('CompanyID').AsString := sCompanyID;
      ParamByName('SupplierID').AsString := DataSet.FieldByName('SupplierID').NewValue;
      ParamByName('LastPurchaseDate').AsString := DataSet.FieldByName('PurchaseDate').NewValue;
      ExecSQL;
    end;
  end;
end;

procedure TfmPR110.sbSelectClick(Sender: TObject);
begin
  inherited;
  ReData := False;
  qyMaster.DisableControls;
  try
    fmPR110Select := TfmPR110Select.Create(Application);
    fmPR110Select.ShowModal;
    if ReData then begin
      try
        fmLoading := TfmLoading.Create(Application);
        fmLoading.Show;
        fmLoading.Update;
        qyMaster.Close;
        qyMaster.Open;
      finally
        fmLoading.Hide;
        fmLoading.Free;
      end;
    end;
  finally
```

```
  qyMaster.EnableControls;
  fmPR110Select.Free;
 end;
end;

procedure TfmPR110.sbReportClick(Sender: TObject);
begin
 inherited;
 PR110Print(qyMaster.FieldByName('PurchaseID').AsString);
end;
```




```
unit DL110;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  MasterDetail, Menus, Db, DBTables, Grids, DBGrids, Buttons, ExtCtrls,
  StdCtrls, Mask, DBCtrls;

type
  TfmDL110 = class(TfmMasterDetail)
```

```
qyMasterCompanyID: TStringField;
qyMasterDeliveryID: TStringField;
qyMasterDeliveryDate: TStringField;
qyMasterCustomerID: TStringField;
qyMasterCustomerAttribName: TStringField;
qyMasterWarehouseID: TStringField;
qyMasterWarehouseName: TStringField;
qyMasterSalesManID: TStringField;
qyMasterChineseName: TStringField;
qyMasterDeliveryProperty: TStringField;
qyMasterInvoiceType: TStringField;
qyMasterInvoiceNo: TStringField;
qyMasterCustomerOrderNo: TStringField;
qyMasterSubTotal: TBCDField;
qyMasterValueAddTax: TBCDField;
qyMasterAmount: TBCDField;
qyMasterSales: TBCDField;
qyMasterTax: TBCDField;
qyMasterReceived: TBCDField;
qyMasterCreateMan: TStringField;
qyMasterCreateDate: TStringField;
qyMasterCreateTime: TStringField;
qyMasterModifyMan: TStringField;
qyMasterModifyDate: TStringField;
qyMasterModifyTime: TStringField;
Label1: TLabel;
DBEdit1: TDBEdit;
Label2: TLabel;
Label3: TLabel;
ED2: TDBEdit;
DBEdit4: TDBEdit;
Label5: TLabel;
ED5: TDBEdit;
DBEdit6: TDBEdit;
Label7: TLabel;
ED4: TDBEdit;
DBEdit8: TDBEdit;
sbWarehouseID: TSpeedButton;
sbSalesManID: TSpeedButton;
sbCustomerID: TSpeedButton;
Label13: TLabel;
Label14: TLabel;
Label15: TLabel;
Label11: TLabel;
Label12: TLabel;
EDC: TDBEdit;
```

```
EDD: TDBEdit;
DBEdit13: TDBEdit;
DBEdit14: TDBEdit;
DBEdit15: TDBEdit;
qyDetailCompanyID: TStringField;
qyDetailDeliveryItem: TStringField;
qyDetailProductID: TStringField;
qyDetailProductName: TStringField;
qyDetailSalesQuantity: TBCDField;
qyDetailSalesUnitPrice: TBCDField;
qyDetailSalesAmount: TBCDField;
qyDetailSalesAnalystAmount: TBCDField;
sbDeliveryDate: TSpeedButton;
qyMasterDeliveryAddressID: TStringField;
qyMasterAddressDescription: TStringField;
Label4: TLabel;
ED3: TDBEdit;
DBEdit5: TDBEdit;
sbDeliveryAddressID: TSpeedButton;
qyDetailDeliveryID: TStringField;
qyTemp2: TQuery;
qyTemp1: TQuery;
qyCustomerAddress: TQuery;
StringField3: TStringField;
StringField4: TStringField;
edProductID: TDBEdit;
rgDeliveryProperty: TDBRadioGroup;
rgInvoiceType: TDBRadioGroup;
qyMasterAccountReceivable: TBCDField;
procedure FormCreate(Sender: TObject);
procedure qyMasterBeforeOpen(DataSet: TDataSet);
procedure qyMasterNewRecord(DataSet: TDataSet);
procedure qyMasterBeforePost(DataSet: TDataSet);
procedure qyDetailBeforeOpen(DataSet: TDataSet);
procedure qyDetailNewRecord(DataSet: TDataSet);
procedure qyDetailBeforePost(DataSet: TDataSet);
procedure sbDeliveryDateClick(Sender: TObject);
procedure qyDetailSalesQuantityValidate(Sender: TField);
procedure qyDetailSalesUnitPriceValidate(Sender: TField);
procedure dsMasterStateChange(Sender: TObject);
procedure qyMasterCustomerIDValidate(Sender: TField);
procedure qyMasterDeliveryAddressIDValidate(Sender: TField);
procedure qyMasterSalesManIDValidate(Sender: TField);
procedure qyMasterWarehouseIDValidate(Sender: TField);
procedure qyDetailProductIDValidate(Sender: TField);
procedure gdDetailEditButtonClick(Sender: TObject);
```

```
    procedure sbSalesManIDClick(Sender: TObject);
    procedure sbCustomerIDClick(Sender: TObject);
    procedure sbDeliveryAddressIDClick(Sender: TObject);
    procedure qyCustomerAddressBeforeOpen(DataSet: TDataSet);
    procedure sbWarehouseIDClick(Sender: TObject);
    procedure qyDetailAfterInsert(DataSet: TDataSet);
    procedure qyDetailUpdateRecord(DataSet: TDataSet;
      UpdateKind: TUpdateKind; var UpdateAction: TUpdateAction);
    procedure qyMasterUpdateRecord(DataSet: TDataSet;
      UpdateKind: TUpdateKind; var UpdateAction: TUpdateAction);
    procedure sbSelectClick(Sender: TObject);
    procedure sbReportClick(Sender: TObject);
  private
    { Private declarations }
    procedure CalcSalesAmount;
    procedure UpdateSalesAnalyst(DeliveryDate, CustomerID, ProductID,
          DeliveryProperty: String; SalesQuantity, SalesAmount: Extended);
    procedure UpdateCustomerInventory(CustomerID, ProductID: String;
          Quantity: Extended);
  public
    { Public declarations }
    AStartPeriodDate, AEndPeriodDate : String;
    AStartCustomerID, AEndCustomerID : String;
    AStartSalesManID, AEndSalesManID : String;
    AStartWarehouseID, AEndWarehouseID : String;
    AProperty1, AProperty2, AProperty3, AProperty4 : String;
    ReData : Boolean;
    procedure OpenDB; override;
    procedure CalcAmount; override;
    procedure GetPrevValue; override;
  end;

var
  fmDL110: TfmDL110;

implementation

uses Main, DataModule, PublicFunction, CheckData, GetData,
  BaseSearch, DL110Select, Loading, DL110Report, CalendarSearch;

var
  sPrevDeliveryDate, sPrevCustomerID, sPrevWarehouseID, sPrevDeliveryProperty : String;

{$R *.DFM}

{ TfmDL110 }
```

```
procedure TfmDL110.FormCreate(Sender: TObject);
begin

  AStartPeriodDate := sStartPeriodDate;
  AEndPeriodDate := sEndPeriodDate;
  AStartCustomerID := '00000000';
  AEndCustomerID := 'ZZZZZZZZ';
  AStartSalesManID := '0000';
  AEndSalesManID := 'ZZZZ';
  AStartWarehouseID := '0000';
  AEndWarehouseID := 'ZZZZ';
  AProperty1 := '1';
  AProperty2 := '2';
  AProperty3 := '3';
  AProperty4 := '4';
  ReData := False;
  inherited;

  sPrevDeliveryDate := '';
  sPrevCustomerID := '';
  sPrevWarehouseID := '';
  sPrevDeliveryProperty := '';

end;

procedure TfmDL110.OpenDB;
begin
  qyMaster.Close;
  OrderBySQL := 'M.DeliveryID';
  qyMaster.SQL.Text := SQLText;
  qyMaster.Open;
  ItemFieldName := 'DeliveryItem';
  DataDate := 'DeliveryDate';

end;

procedure TfmDL110.CalcAmount;
var
  fSubTotal : Extended;
  PrevBookmark : TBookmark;
begin

  fSubTotal := 0;
  PrevBookmark := qyDetail.GetBookmark;
  try
```

```
    qyDetail.DisableControls;
    qyDetail.First;
    while not qyDetail.Eof do
    begin
      fSubTotal := fSubTotal + qyDetail.FieldByName('SalesAmount').AsFloat;
      qyDetail.Next;
    end;
    qyMaster.FieldByName('SubTotal').AsFloat := fSubTotal;
    if qyMaster.FieldByName('InvoiceType').AsString = '3' then
      qyMaster.FieldByName('ValueAddTax').AsFloat :=
          RealToInt(qyMaster.FieldByName('SubTotal').AsFloat * 0.05)
    else
      qyMaster.FieldByName('ValueAddTax').AsFloat := 0;
    qyMaster.FieldByName('Amount').AsFloat :=
        qyMaster.FieldByName('SubTotal').AsFloat +
        qyMaster.FieldByName('ValueAddTax').AsFloat;
  finally
    qyDetail.EnableControls;
    if PrevBookmark <> nil then
    begin
      qyDetail.GotoBookmark(PrevBookmark);
      qyDetail.FreeBookmark(PrevBookmark);
    end;
  end;
end;

procedure TfmDL110.qyMasterBeforeOpen(DataSet: TDataSet);
begin
  inherited;

  with qyMaster do
  begin
    ParamByName('StartPeriodDate').AsString := AStartPeriodDate;
    ParamByName('EndPeriodDate').AsString := AEndPeriodDate;
    ParamByName('StartCustomerID').AsString := AStartCustomerID;
    ParamByName('EndCustomerID').AsString := AEndCustomerID;
    ParamByName('StartSalesManID').AsString := AStartSalesManID;
    ParamByName('EndSalesManID').AsString := AEndSalesManID;
    ParamByName('StartWarehouseID').AsString := AStartWarehouseID;
    ParamByName('EndWarehouseID').AsString := AEndWarehouseID;
    ParamByName('Property1').AsString := AProperty1;
    ParamByName('Property2').AsString := AProperty2;
    ParamByName('Property3').AsString := AProperty3;
    ParamByName('Property4').AsString := AProperty4;
  end;
end;
```

```
procedure TfmDL110.qyMasterNewRecord(DataSet: TDataSet);
begin
 inherited;

 with qyMaster do
 begin
  FieldByName('DeliveryID').AsString := Space(12);
  FieldByName('DeliveryDate').AsString := Today;
  FieldByName('CustomerID').AsString := Space(8);
  FieldByName('DeliveryAddressID').AsString := Space(2);
  FieldByName('WarehouseID').AsString := Space(2);
  FieldByName('SalesManID').AsString := Space(4);
  FieldByName('DeliveryProperty').AsString := '1';
  FieldByName('InvoiceType').AsString := '3';
  FieldByName('InvoiceNo').AsString := Space(8);
  FieldbyName('CustomerOrderNo').AsString := Space(15);
  FieldByName('SubTotal').AsFloat := 0;
  FieldByName('ValueAddTax').AsFloat := 0;
  FieldByName('Amount').AsFloat := 0;
  FieldByName('Sales').AsFloat := 0;
  FieldByName('Tax').AsFloat := 0;
  FieldByName('AccountReceivable').AsFloat := 0;
  FieldByName('Received').AsFloat := 0;
 end;
end;

procedure TfmDL110.qyMasterBeforePost(DataSet: TDataSet);
begin
 inherited;

 if qyMaster.State = dsInsert then
 begin
  if not CheckDate(qyMaster.FieldByName('DeliveryDate').AsString) then
  begin
   ED1.SetFocus;
   Abort;
  end;
  if not CheckCustomerID(qyMaster.FieldByName('CustomerID').AsString) then
  begin
   ED2.SetFocus;
   Abort;
  end;
 end;
 if not CheckAddressID(qyMaster.FieldByName('CustomerID').AsString,
          qyMaster.FieldByName('DeliveryAddressID').AsString) then
```

```
 begin
  ED3.SetFocus;
  Abort;
 end;
 if not CheckSalesManID(qyMaster.FieldByName('SalesManID').AsString) then
 begin
  ED4.SetFocus;
  Abort;
 end;
 if not CheckWarehouseID(qyMaster.FieldByName('WarehouseID').AsString) then
 begin
  ED5.SetFocus;
  Abort;
 end;
 if qyMaster.State = dsInsert then
  qyMaster.FieldByName('DeliveryID').AsString :=
        IDGen('DL', qyMaster.FieldByName('DeliveryDate').AsString,
          'DeliveryID', 'DeliveryMaster');
 if qyMaster.FieldByName('InvoiceType').AsString = '3' then
 begin
  qyMaster.FieldByName('Sales').AsFloat := qyMaster.FieldByName('SubTotal').AsFloat;
  qyMaster.FieldByName('Tax').AsFloat := qyMaster.FieldByName('ValueAddTax').AsFloat;
 end
 else
 begin
  qyMaster.FieldByName('Sales').AsFloat := qyMaster.FieldByName('Amount').AsFloat / 1.05;
  qyMaster.FieldByName('Tax').AsFloat := qyMaster.FieldByName('Amount').AsFloat -
                        qyMaster.FieldByName('Sales').AsFloat;
 end;
 if (qyMaster.FieldByName('DeliveryProperty').AsString = '1') then
  qyMaster.FieldByName('AccountReceivable').AsFloat :=
       qyMaster.FieldByName('Amount').AsFloat
 else
  if (qyMaster.FieldByName('DeliveryProperty').AsString = '2') then
   qyMaster.FieldByName('AccountReceivable').AsFloat :=
         qyMaster.FieldByName('Amount').AsFloat * -1
  else
   qyMaster.FieldByName('AccountReceivable').AsFloat := 0;
end;

procedure TfmDL110.qyDetailBeforeOpen(DataSet: TDataSet);
begin
 inherited;

 qyDetail.ParamByName('DeliveryID').AsString := qyMaster.FieldByName('DeliveryID').AsString;
end;
```

```
procedure TfmDL110.qyDetailNewRecord(DataSet: TDataSet);
begin
 inherited;

 with qyDetail do
 begin
  FieldByName('DeliveryID').AsString := qyMaster.FieldByName('DeliveryID').AsString;
  FieldByName('DeliveryItem').AsString := Space(3);
  FieldByName('ProductID').AsString := Space(10);
  FieldByName('SalesQuantity').AsFloat := 0;
  FieldByName('SalesUnitPrice').AsFloat := 0;
  FieldByName('SalesAmount').AsFloat := 0;
  FieldByName('SalesAnalystAmount').AsFloat := 0;
 end;
end;

procedure TfmDL110.qyDetailBeforePost(DataSet: TDataSet);
begin
 inherited;

 if not CheckProductID(qyDetail.FieldByName('ProductID').AsString) then
 begin
  gdDetail.SelectedField := qyDetailProductID;
  Abort;
 end;
 CalcSalesAmount;
 if qyMaster.FieldByName('InvoiceType').AsString = '3' then
  qyDetail.FieldByName('SalesAnalystAmount').AsFloat :=
      qyDetail.FieldByName('SalesAmount').AsFloat
 else
  qyDetail.FieldByName('SalesAnalystAmount').AsFloat :=
      RealToInt(qyDetail.FieldByName('SalesAmount').AsFloat / 1.05);
 if qyDetail.State = dsInsert then
  qyDetail.FieldByName('DeliveryItem').AsString :=
      ITGen(qyMaster.FieldByName('DeliveryID').AsString,
        'DeliveryItem', 'DeliveryID', 'DeliveryDetail');
end;

procedure TfmDL110.sbDeliveryDateClick(Sender: TObject);
begin
 inherited;
// MyCalendar(ED1);
 ShowCalendar(ED1);
end;
```

```
procedure TfmDL110.qyDetailSalesQuantityValidate(Sender: TField);
begin
  inherited;
  CalcSalesAmount;
end;

procedure TfmDL110.qyDetailSalesUnitPriceValidate(Sender: TField);
begin
  inherited;
  CalcSalesAmount;
end;

procedure TfmDL110.CalcSalesAmount;
begin
  qyDetail.FieldByName('SalesAmount').AsFloat :=
        qyDetail.FieldByName('SalesQuantity').AsFloat *
        qyDetail.FieldByName('SalesUnitPrice').AsFloat;
end;

procedure TfmDL110.dsMasterStateChange(Sender: TObject);
begin
  inherited;
  ED2.ReadOnly := (qyMaster.State <> dsInsert);
  ED5.ReadOnly := (qyMaster.State <> dsInsert);
  sbDeliveryDate.Enabled := (qyMaster.State = dsInsert);
  rgDeliveryProperty.ReadOnly := (qyMaster.State <> dsInsert);
  rgInvoiceType.ReadOnly := (qyMaster.State <> dsInsert);
  sbCustomerID.Enabled := (qyMaster.State = dsInsert);
  sbDeliveryAddressID.Enabled := (qyMaster.State in [dsInsert, dsEdit]);
  sbSalesManID.Enabled := (qyMaster.State in [dsInsert, dsEdit]);
  sbWarehouseID.Enabled := (qyMaster.State = dsInsert);
end;

procedure TfmDL110.UpdateSalesAnalyst(DeliveryDate, CustomerID,
  ProductID, DeliveryProperty: String; SalesQuantity, SalesAmount: Extended);
begin
  with qyTemp do
  begin
    Close;
    SQL.Clear;
    SQL.Add('SELECT CustomerID ');
    SQL.Add('FROM SalesAnalyst ');
    SQL.Add('WHERE CompanyID = :CompanyID ');
    SQL.Add('AND SalesYear = :SalesYear AND SalesMonth = :SalesMonth ');
    SQL.Add('AND CustomerID = :CustomerID AND ProductID = :ProductID ');
    ParamByName('CompanyID').AsString := sCompanyID;
```

```
  ParamByName('SalesYear').AsString := RightStr(DeliveryDate, 2);
  ParamByName('SalesMonth').AsString := SubStr(DeliveryDate, 4, 2);
  ParamByName('CustomerID').AsString := CustomerID;
  ParamByName('ProductID').AsString := ProductID;
  Open;
 end;
 if qyTemp.FieldByName('CustomerID').AsString = '' then
 begin
  with qyTemp2 do
  begin
   Close;
   SQL.Clear;
   SQL.Add('INSERT INTO SalesAnalyst ');
   SQL.Add(' (CompanyID, SalesYear, SalesMonth, CustomerID, ProductID, ');
   SQL.Add('  SalesQuantity, SalesAmount, SalesReturnQuantity, SalesReturnAmount) ');
   SQL.Add('VALUES ');
   SQL.Add(' (:CompanyID, :SalesYear, :SalesMonth, :CustomerID, :ProductID, ');
   SQL.Add('  :SalesQuantity, :SalesAmount, :SalesReturnQuantity, :SalesReturnAmount) ');
   ParamByName('CompanyID').AsString := sCompanyID;
   ParamByName('SalesYear').AsString := RightStr(DeliveryDate, 2);
   ParamByName('SalesMonth').AsString := SubStr(DeliveryDate, 4, 2);
   ParamByName('CustomerID').AsString := CustomerID;
   ParamByName('ProductID').AsString := ProductID;
   if DeliveryProperty = '1' then
   begin
    ParamByName('SalesQuantity').AsFloat := SalesQuantity;
    ParamByName('SalesAmount').AsFloat := SalesAmount;
    ParamByName('SalesReturnQuantity').AsFloat := 0;
    ParamByName('SalesReturnAmount').AsFloat := 0;
   end
   else
   begin
    ParamByName('SalesQuantity').AsFloat := 0;
    ParamByName('SalesAmount').AsFloat := 0;
    ParamByName('SalesReturnQuantity').AsFloat := SalesQuantity;
    ParamByName('SalesReturnAmount').AsFloat := SalesAmount;
   end;
   ExecSQL;
  end;
 end
 else
 begin
  with qyTemp2 do
  begin
   Close;
   SQL.Clear;
```

```
    SQL.Add('UPDATE SalesAnalyst ');
    SQL.Add('SET SalesQuantity = SalesQuantity + :SalesQuantity, ');
    SQL.Add('    SalesAmount = SalesAmount + :SalesAmount, ');
    SQL.Add('    SalesReturnQuantity = SalesReturnQuantity + :SalesReturnQuantity, ');
    SQL.Add('    SalesReturnAmount = SalesReturnAmount + :SalesReturnAmount ');
    SQL.Add('WHERE CompanyID = :CompanyID ');
    SQL.Add('AND SalesYear = :SalesYear AND SalesMonth = :SalesMonth ');
    SQL.Add('AND CustomerID = :CustomerID AND ProductID = :ProductID ');
    ParamByName('CompanyID').AsString := sCompanyID;
    ParamByName('SalesYear').AsString := RightStr(DeliveryDate, 2);
    ParamByName('SalesMonth').AsString := SubStr(DeliveryDate, 4, 2);
    ParamByName('CustomerID').AsString := CustomerID;
    ParamByName('ProductID').AsString := ProductID;
    if DeliveryProperty = '1' then
    begin
      ParamByName('SalesQuantity').AsFloat := SalesQuantity;
      ParamByName('SalesAmount').AsFloat := SalesAmount;
      ParamByName('SalesReturnQuantity').AsFloat := 0;
      ParamByName('SalesReturnAmount').AsFloat := 0;
    end
    else
    begin
      ParamByName('SalesQuantity').AsFloat := 0;
      ParamByName('SalesAmount').AsFloat := 0;
      ParamByName('SalesReturnQuantity').AsFloat := SalesQuantity;
      ParamByName('SalesReturnAmount').AsFloat := SalesAmount;
    end;
    ExecSQL;
  end;
 end;
end;

procedure TfmDL110.UpdateCustomerInventory(CustomerID, ProductID: String;
 Quantity: Extended);
begin
 with qyTemp do
 begin
   Close;
   SQL.Clear;
   SQL.Add('SELECT CustomerID ');
   SQL.Add('FROM CustomerInventory ');
   SQL.Add('WHERE CompanyID = :CompanyID ');
   SQL.Add('AND CustomerID = :CustomerID AND ProductID = :ProductID ');
   ParamByName('CompanyID').AsString := sCompanyID;
   ParamByName('CustomerID').AsString := CustomerID;
   ParamByName('ProductID').AsString := ProductID;
```

```
  Open;
 end;
 if qyTemp.FieldByName('CustomerID').AsString = '' then
 begin
  with qyTemp2 do
  begin
   Close;
   SQL.Clear;
   SQL.Add('INSERT INTO CustomerInventory ');
   SQL.Add(' (CompanyID, CustomerID, ProductID, Quantity)');
   SQL.Add('VALUES ');
   SQL.Add(' (:CompanyID, :CustomerID, :ProductID, :Quantity) ');
   ParamByName('CompanyID').AsString := sCompanyID;
   ParamByName('CustomerID').AsString := CustomerID;
   ParamByName('ProductID').AsString := ProductID;
   ParamByName('Quantity').AsFloat := Quantity;
   ExecSQL;
  end;
 end
 else
 begin
  with qyTemp2 do
  begin
   Close;
   SQL.Clear;
   SQL.Add('UPDATE CustomerInventory ');
   SQL.Add('SET Quantity = Quantity + :Quantity ');
   SQL.Add('WHERE CompanyID = :CompanyID ');
   SQL.Add('AND CustomerID = :CustomerID AND ProductID = :ProductID ');
   ParamByName('CompanyID').AsString := sCompanyID;
   ParamByName('CustomerID').AsString := CustomerID;
   ParamByName('ProductID').AsString := ProductID;
   ParamByName('Quantity').AsFloat := Quantity;
   ExecSQL;
  end;
 end;
end;

procedure TfmDL110.qyMasterCustomerIDValidate(Sender: TField);
begin
 inherited;
 with qyTemp do
 begin
  Close;
  SQL.Clear;
  SQL.Add('SELECT CustomerAttribName, SalesManID, DeliveryAddressID, InvoiceType ');
```

```
  SQL.Add('FROM CustomerMaster ');
  SQL.Add('WHERE CompanyID = :CompanyID ');
  SQL.Add('AND CustomerID = :CustomerID ');
  ParamByName('CompanyID').AsString := sCompanyID;
  ParamByName('CustomerID').AsString := qyMaster.FieldByName('CustomerID').AsString;
  Open;
 end;
 qyMaster.FieldByName('CustomerAttribName').AsString :=
      qyTemp.FieldByName('CustomerAttribName').AsString;
 if Trim(qyMaster.FieldByName('SalesManID').AsString) = '' then
  qyMaster.FieldByName('SalesManID').AsString :=
       qyTemp.FieldByName('SalesManID').AsString;
 if Trim(qyMaster.FieldByName('DeliveryAddressID').AsString) = '' then
  qyMaster.FieldByName('DeliveryAddressID').AsString :=
       qyTemp.FieldByName('DeliveryAddressID').AsString;
 qyMaster.FieldByName('InvoiceType').AsString :=
      qyTemp.FieldByName('InvoiceType').AsString;
end;

procedure TfmDL110.qyMasterDeliveryAddressIDValidate(Sender: TField);
begin
 inherited;
 qyMaster.FieldByName('AddressDescription').AsString :=
      GetAddressDescription(qyMaster.FieldByName('CustomerID').AsString,
                qyMaster.FieldByName('DeliveryAddressID').AsString);
end;

procedure TfmDL110.qyMasterSalesManIDValidate(Sender: TField);
begin
 inherited;
 qyMaster.FieldByName('ChineseName').AsString :=
      GetSalesName(qyMaster.FieldByName('SalesManID').AsString);
end;

procedure TfmDL110.qyMasterWarehouseIDValidate(Sender: TField);
begin
 inherited;
 qyMaster.FieldByName('WarehouseName').AsString :=
      GetWarehouseName(qyMaster.FieldByName('WarehouseID').AsString);
end;

procedure TfmDL110.qyDetailProductIDValidate(Sender: TField);
begin
 inherited;
 qyDetail.FieldByName('ProductName').AsString :=
      GetProductName(qyDetail.FieldByName('ProductID').AsString);
```

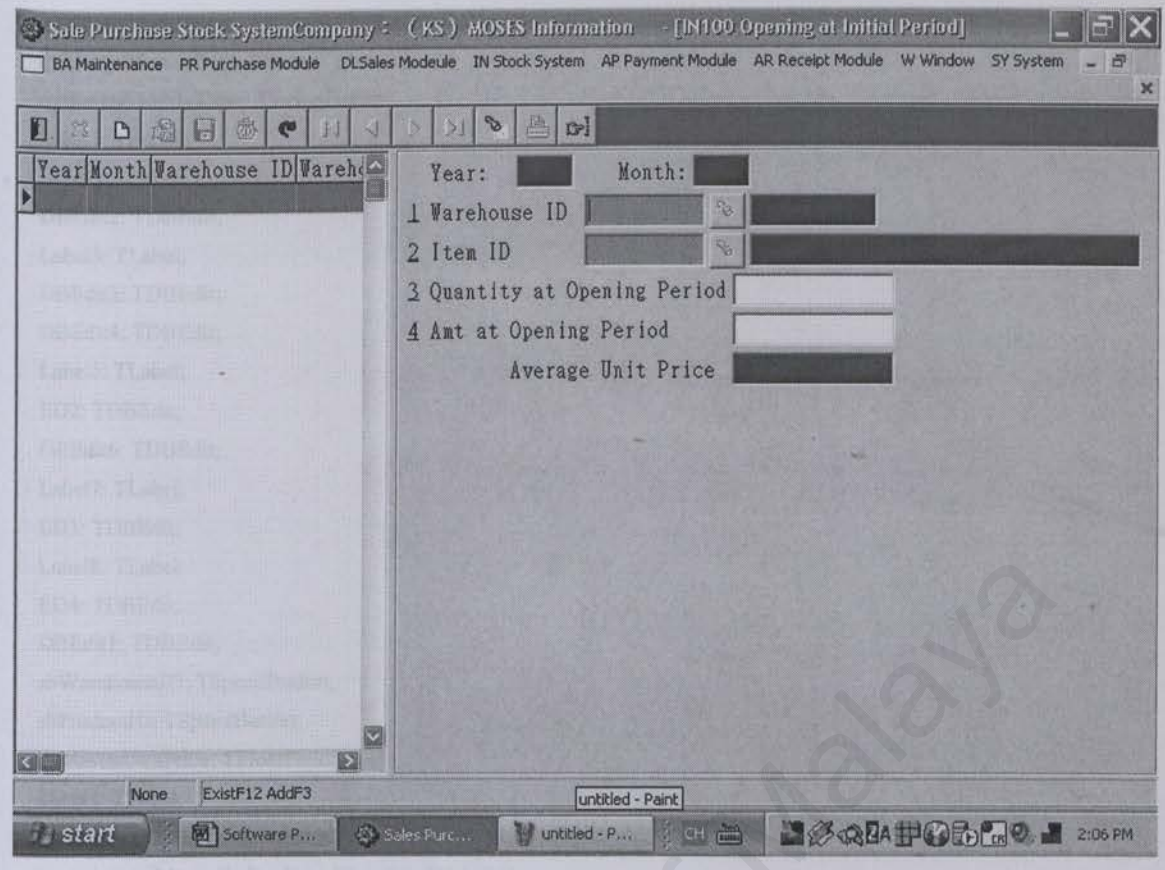
```
end;

procedure TfmDL110.gdDetailEditButtonClick(Sender: TObject);
begin
  inherited;
  if (gdDetail.SelectedField.FieldName = 'ProductID') and
     (qyDetail.State in [dsInsert, dsEdit]) then
    qyDetail.FieldByName('ProductID').AsString := SearchData(edProductID, DM.qyProduct);
end;

procedure TfmDL110.sbSalesManIDClick(Sender: TObject);
begin
  inherited;
  SearchData(ED4, DM.qySalesMan);
end;

procedure TfmDL110.sbCustomerIDClick(Sender: TObject);
begin
  inherited;
  SearchData(ED2, DM.qyCustomerMaster);
end;

procedure TfmDL110.sbDeliveryAddressIDClick(Sender: TObject);
begin
  inherited;
  SearchData(ED3, qyCustomerAddress);
end;

procedure TfmDL110.qyCustomerAddressBeforeOpen(DataSet: TDataSet);
begin
  inherited;
  with qyCustomerAddress do
  begin
    ParamByName('CompanyID').AsString := sCompanyID;
    ParamByName('CustomerID').AsString := qyMaster.FieldByName('CustomerID').AsString;
  end;
end;

procedure TfmDL110.sbWarehouseIDClick(Sender: TObject);
begin
  inherited;
  SearchData(ED5, DM.qyWarehouse);
end;

procedure TfmDL110.qyDetailAfterInsert(DataSet: TDataSet);
begin
```

```
 inherited;
 gdDetail.SelectedField := qyDetailProductID;
end;

procedure TfmDL110.qyDetailUpdateRecord(DataSet: TDataSet;
 UpdateKind: TUpdateKind; var UpdateAction: TUpdateAction);
var
 fNewSalesQuantity, fOldSalesQuantity : Extended;
 fNewSalesAnalystAmount, fOldSalesAnalystAmount : Extended;
 sNewProductID, sOldProductID : String;
begin
 inherited;
 if ReSortting then Exit;
 fNewSalesQuantity := 0;
 fOldSalesQuantity := 0;
 fNewSalesAnalystAmount := 0;
 fOldSalesAnalystAmount := 0;
 sNewProductID := '';
 sOldProductID := '';
 case UpdateKind of
  ukInsert :
   begin
    fNewSalesQuantity := DataSet.FieldByName('SalesQuantity').NewValue;
    fOldSalesQuantity := 0;
    fNewSalesAnalystAmount := DataSet.FieldByName('SalesAnalystAmount').NewValue;
    fOldSalesAnalystAmount := 0;
    sNewProductID := DataSet.FieldByName('ProductID').NewValue;
    sOldProductID := DataSet.FieldByName('ProductID').NewValue;
    with DM.qyTemp1 do
    begin
     Close;
     SQL.Clear;
     SQL.Add('UPDATE Product SET LastDeliveryDate = :LastDeliveryDate ');
     SQL.Add('WHERE CompanyID = :CompanyID AND ProductID = :ProductID ');
     ParamByName('CompanyID').AsString := sCompanyID;
     ParamByName('ProductID').AsString := DataSet.FieldByName('ProductID').AsString;
     ParamByName('LastDeliveryDate').AsString := qyMaster.FieldByName('DeliveryDate').AsString;
     ExecSQL;
    end;
   end;
  ukModify :
   begin
    fNewSalesQuantity := DataSet.FieldByName('SalesQuantity').NewValue;
    fOldSalesQuantity := DataSet.FieldByName('SalesQuantity').OldValue;
    fNewSalesAnalystAmount := DataSet.FieldByName('SalesAnalystAmount').NewValue;
    fOldSalesAnalystAmount := DataSet.FieldByName('SalesAnalystAmount').OldValue;
```

```
    sNewProductID := DataSet.FieldByName('ProductID').NewValue;
    sOldProductID := DataSet.FieldByName('ProductID').OldValue;
   end;
  ukDelete :
   begin
    fNewSalesQuantity := 0;
    fOldSalesQuantity := DataSet.FieldByName('SalesQuantity').OldValue;
    fNewSalesAnalystAmount := 0;
    fOldSalesAnalystAmount := DataSet.FieldByName('SalesAnalystAmount').OldValue;
    sNewProductID := DataSet.FieldByName('ProductID').OldValue;
    sOldProductID := DataSet.FieldByName('ProductID').OldValue;
   end;
end;
if (sPrevDeliveryProperty = '1') or (sPrevDeliveryProperty = '2') then
begin
 UpdateSalesAnalyst(sPrevDeliveryDate, sPrevCustomerID,
            sOldProductID, sPrevDeliveryProperty,
            (fOldSalesQuantity * -1),
            (fOldSalesAnalystAmount * -1));
 UpdateSalesAnalyst(sPrevDeliveryDate, sPrevCustomerID,
            sNewProductID, sPrevDeliveryProperty,
            fNewSalesQuantity,
            fNewSalesAnalystAmount);
end;
if (sPrevDeliveryProperty = '3') then
begin
 UpdateCustomerInventory(sPrevCustomerID, sOldProductID,
            (fOldSalesQuantity * -1));
 UpdateCustomerInventory(sPrevCustomerID, sNewProductID,
            fNewSalesQuantity);
end;
if (sPrevDeliveryProperty = '4') then
begin
 UpdateCustomerInventory(sPrevCustomerID, sOldProductID,
            fOldSalesQuantity);
 UpdateCustomerInventory(sPrevCustomerID, sNewProductID,
            (fNewSalesQuantity * -1));
end;
UpdateInventoryAnalyst(sPrevDeliveryDate, sOldProductID,
            sPrevWarehouseID, '0' + sPrevDeliveryProperty,
            (fOldSalesQuantity * -1),
            (fOldSalesAnalystAmount * -1));
UpdateInventoryAnalyst(sPrevDeliveryDate, sNewProductID,
            sPrevWarehouseID, '0' + sPrevDeliveryProperty,
            fNewSalesQuantity,
            fNewSalesAnalystAmount);
```

```
end;

procedure TfmDL110.GetPrevValue;
begin
 inherited;
 sPrevDeliveryDate := qyMaster.FieldByName('DeliveryDate').AsString;
 sPrevCustomerID := qyMaster.FieldByName('CustomerID').AsString;
 sPrevWarehouseID := qyMaster.FieldByName('WarehouseID').AsString;
 sPrevDeliveryProperty := qyMaster.FieldByName('DeliveryProperty').AsString;
end;

procedure TfmDL110.qyMasterUpdateRecord(DataSet: TDataSet;
 UpdateKind: TUpdateKind; var UpdateAction: TUpdateAction);
var
 fNewAmount, fOldAmount : Extended;
 sLastDeliveryDate, sCustomerID : String;
begin
 inherited;
 fNewAmount := 0;
 fOldAmount := 0;
 sLastDeliveryDate := '';
 case UpdateKind of
  ukInsert :
   begin
    fNewAmount := DataSet.FieldByName('AccountReceivable').NewValue;
    fOldAmount := 0;
    sLastDeliveryDate := DataSet.FieldByName('DeliveryDate').NewValue;
    sCustomerID := DataSet.FieldByName('CustomerID').NewValue;
   end;
  ukModify :
   begin
    fNewAmount := DataSet.FieldByName('AccountReceivable').NewValue;
    fOldAmount := DataSet.FieldByName('AccountReceivable').OldValue;
    sLastDeliveryDate := DataSet.FieldByName('DeliveryDate').OldValue;
    sCustomerID := DataSet.FieldByName('CustomerID').OldValue;
   end;
  ukDelete :
   begin
    fNewAmount := DataSet.FieldByName('AccountReceivable').OldValue;
    fOldAmount := DataSet.FieldByName('AccountReceivable').OldValue;
    sLastDeliveryDate := DataSet.FieldByName('DeliveryDate').OldValue;
    sCustomerID := DataSet.FieldByName('CustomerID').OldValue;
   end;
 end;
 with qyTemp do
 begin
```

```
    Close;
    SQL.Clear;
    SQL.Add('UPDATE CustomerMaster SET ');
    SQL.Add('LastDeliveryDate = :LastDeliveryDate, ');
    SQL.Add('CreditBalance = CreditBalance + :CreditBalance ');
    SQL.Add('WHERE CompanyID = :CompanyID AND CustomerID = :CustomerID ');
    ParamByName('CompanyID').AsString := sCompanyID;
    ParamByName('CustomerID').AsString := sCustomerID;
    ParamByName('LastDeliveryDate').AsString := sLastDeliveryDate;
    ParamByName('CreditBalance').AsFloat := fOldAmount - fNewAmount;
    ExecSQL;
  end;
end;

procedure TfmDL110.sbSelectClick(Sender: TObject);
begin
  inherited;
  ReData := False;
  qyMaster.DisableControls;
  try
    fmDL110Select := TfmDL110Select.Create(Application);
    fmDL110Select.ShowModal;
    if ReData then begin
      try
        fmLoading := TfmLoading.Create(Application);
        fmLoading.Show;
        fmLoading.Update;
        qyMaster.Close;
        qyMaster.Open;
      finally
        fmLoading.Hide;
        fmLoading.Free;
      end;
    end;
  finally
    qyMaster.EnableControls;
    fmDL110Select.Free;
  end;
end;

procedure TfmDL110.sbReportClick(Sender: TObject);
begin
  inherited;
  DL110Print(qyMaster.FieldByName('DeliveryID').AsString);
end;
end.
```




```
unit IN100;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Single, Menus, Db, DBTables, Grids, DBGrids, Buttons, ExtCtrls, StdCtrls,
  Mask, DBCtrls;

type
  TfmIN100 = class(TfmSingle)
    qyMasterCompanyID: TStringField;
    qyMasterInventoryYear: TStringField;
    qyMasterInventoryMonth: TStringField;
    qyMasterWarehouseID: TStringField;
    qyMasterWarehouseName: TStringField;
    qyMasterProductID: TStringField;
    qyMasterProductName: TStringField;
    qyMasterBeginningQuantity: TBCDField;
    qyMasterBeginningBalance: TBCDField;
    qyMasterCreateMan: TStringField;
    qyMasterCreateDate: TStringField;
    qyMasterCreateTime: TStringField;
```

```
    qyMasterModifyMan: TStringField;
    qyMasterModifyDate: TStringField;
    qyMasterModifyTime: TStringField;
    Label1: TLabel;
    Label2: TLabel;
    DBEdit2: TDBEdit;
    Label3: TLabel;
    DBEdit3: TDBEdit;
    DBEdit4: TDBEdit;
    Label5: TLabel;
    ED2: TDBEdit;
    DBEdit6: TDBEdit;
    Label7: TLabel;
    ED3: TDBEdit;
    Label8: TLabel;
    ED4: TDBEdit;
    DBEdit1: TDBEdit;
    sbWarehouseID: TSpeedButton;
    sbProductID: TSpeedButton;
    qyMasterUnitPrice: TFloatField;
    Label4: TLabel;
    DBEdit5: TDBEdit;
    procedure qyMasterBeforeOpen(DataSet: TDataSet);
    procedure qyMasterNewRecord(DataSet: TDataSet);
    procedure qyMasterBeforePost(DataSet: TDataSet);
    procedure qyMasterCalcFields(DataSet: TDataSet);
    procedure qyMasterBeforeDelete(DataSet: TDataSet);
    procedure FormCreate(Sender: TObject);
    procedure dsMasterStateChange(Sender: TObject);
    procedure sbWarehouseIDClick(Sender: TObject);
    procedure sbProductIDClick(Sender: TObject);
    procedure qyMasterWarehouseIDValidate(Sender: TField);
    procedure qyMasterProductIDValidate(Sender: TField);
    procedure qyMasterBeforeEdit(DataSet: TDataSet);
    procedure qyMasterBeforeInsert(DataSet: TDataSet);
    procedure sbSelectClick(Sender: TObject);
  private

    { Private declarations }
  public
    { Public declarations }
    AInventoryYear, AInventoryMonth : String;
    AStartWarehouseID, AEndWarehouseID : String;
    ReData : Boolean;
    procedure OpenDB; override;
  end;
```

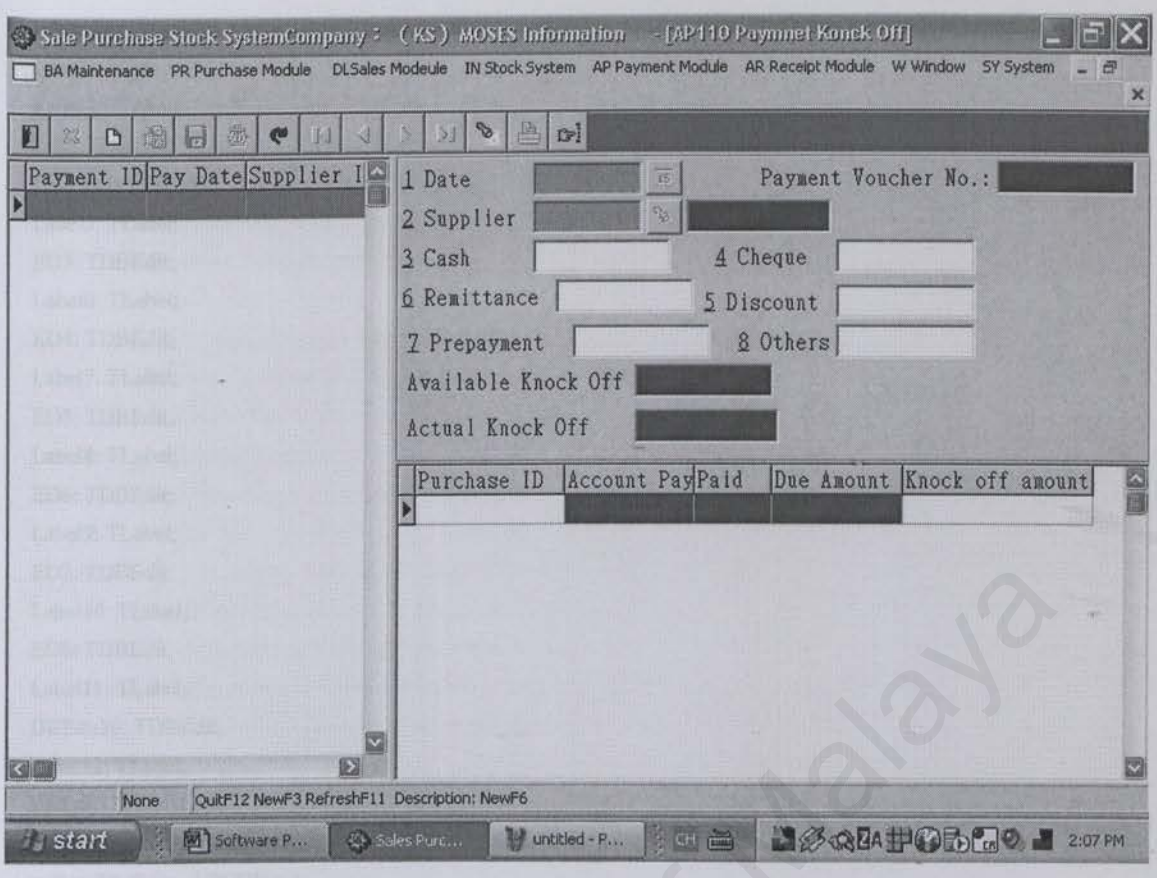
```
var
  fmIN100: TfmIN100;

implementation

uses Main, DataModule, PublicFunction, CheckData, GetData, BaseSearch,
  IN100Select, Loading;

{$R *.DFM}

{ TfmIN100 }

procedure TfmIN100.FormCreate(Sender: TObject);
begin

  AInventoryYear := RightStr(sStartPeriodDate, 2);
  AInventoryMonth := SubStr(sStartPeriodDate, 4, 2);
  AStartWarehouseID := '0000';
  AEndWarehouseID := 'ZZZZ';
  ReData := False;
  inherited;
end;

procedure TfmIN100.OpenDB;
begin
  qyMaster.Close;
  OrderBySQL := 'M.WarehouseID, M.ProductID ';
  qyMaster.SQL.Text := SQLText;
  qyMaster.Open;

end;

procedure TfmIN100.qyMasterBeforeOpen(DataSet: TDataSet);
begin
  //inherited;

  with qyMaster do
  begin
    ParamByName('CompanyID').AsString := sCompanyID;
    ParamByName('InventoryYear').AsString := AInventoryYear;
    ParamByName('InventoryMonth').AsString := AInventoryMonth;
  end;
end;

procedure TfmIN100.qyMasterNewRecord(DataSet: TDataSet);
```

```
begin
 inherited;

 with qyMaster do
 begin
  FieldByName('InventoryYear').AsString := RightStr(sOpenningDate, 2);
  FieldByName('InventoryMonth').AsString := SubStr(sOpenningDate, 4, 2);
  FieldByName('WarehouseID').AsString := Space(4);
  FieldByName('ProductID').AsString := Space(10);
  FieldByName('BeginningQuantity').AsFloat := 0;
  FieldByName('BeginningBalance').AsFloat := 0;
 end;
end;

procedure TfmIN100.qyMasterBeforePost(DataSet: TDataSet);
begin
 inherited;

 if qyMaster.State = dsInsert then
 begin
  if not CheckWarehouseID(qyMaster.FieldByName('WarehouseID').AsString) then
  begin
   ED1.SetFocus;
   Abort;
  end;
  if not CheckProductID(qyMaster.FieldByName('ProductID').AsString) then
  begin
   ED2.SetFocus;
   Abort;
  end;
  with qyTemp do
  begin
   Close;
   SQL.Clear;
   SQL.Add('SELECT ProductID ');
   SQL.Add('FROM InventoryBeginning ');
   SQL.Add('WHERE CompanyID = :CompanyID ');
   SQL.Add('AND InventoryYear = :InventoryYear AND InventoryMonth = :InventoryMonth ');
   SQL.Add('AND WarehouseID = :WarehouseID AND ProductID = :ProductID ');
   ParamByName('CompanyID').AsString := sCompanyID;
   ParamByName('InventoryYear').AsString := qyMaster.FieldByName('InventoryYear').AsString;
   ParamByName('InventoryMonth').AsString := qyMaster.FieldByName('InventoryMonth').AsString;
   ParamByName('WarehouseID').AsString := qyMaster.FieldByName('WarehouseID').AsString;
   ParamByName('ProductID').AsString := qyMaster.FieldByName('ProductID').AsString;
   Open;
  end;
```

```
  if qyTemp.FieldByName('ProductID').AsString <> '' then
  begin
   MyWarning('Openning Data Repeated    ' + #10#13 + #10#13 +
        'Year:[' + qyMaster.FieldByName('InventoryYear').AsString + ']  ' +
        'Month:[' + qyMaster.FieldByName('InventoryMonth').AsString + ']' + #10#13 +
        'Warehouse ID:[' + qyMaster.FieldByName('WarehouseID').AsString + ']' +
              qyMaster.FieldByName('WarehouseName').AsString + #10#13 +
        'Item ID:[' + qyMaster.FieldByName('ProductID').AsString + ']' +
              qyMaster.FieldByName('ProductName').AsString);
   ED1.SetFocus;
   Abort;
  end;
 end;
end;

procedure TfmIN100.qyMasterCalcFields(DataSet: TDataSet);
begin
 inherited;

 if qyMaster.FieldByName('BeginningQuantity').AsFloat <> 0 then
  qyMaster.FieldByName('UnitPrice').AsFloat :=
      qyMaster.FieldByName('BeginningBalance').AsFloat /
      qyMaster.FieldByName('BeginningQuantity').AsFloat
 else
  qyMaster.FieldByName('UnitPrice').AsFloat := 0;
end;

procedure TfmIN100.qyMasterBeforeDelete(DataSet: TDataSet);
begin
 if (qyMaster.FieldByName('InventoryYear').AsString + '-' +
   qyMaster.FieldByName('InventoryMonth').AsString + '-01') <> sStartPeriodDate then
 begin
  MyWarning('Openning data not at opening month, cannot be deleted    ');
  Abort;
 end;
 inherited;
end;

procedure TfmIN100.dsMasterStateChange(Sender: TObject);
begin
 inherited;
 if qyMaster.State in [dsInsert, dsEdit] then
 begin
  sbWarehouseID.Enabled := True;
  sbProductID.Enabled := True;
 end
```

```
  else
  begin
    sbWarehouseID.Enabled := False;
    sbProductID.Enabled := False;
  end;
  ED1.ReadOnly := qyMaster.State <> dsInsert;
  ED2.ReadOnly := qyMaster.State <> dsInsert;
end;

procedure TfmIN100.sbWarehouseIDClick(Sender: TObject);
begin
  inherited;
  SearchData(ED1, DM.qyWarehouse);
end;

procedure TfmIN100.sbProductIDClick(Sender: TObject);
begin
  inherited;
  SearchData(ED2, DM.qyProduct);
end;

procedure TfmIN100.qyMasterWarehouseIDValidate(Sender: TField);
begin
  inherited;
  qyMaster.FieldByName('WarehouseName').AsString :=
        GetWarehouseName(qyMaster.FieldByName('WarehouseID').AsString);
end;

procedure TfmIN100.qyMasterProductIDValidate(Sender: TField);
begin
  inherited;
  qyMaster.FieldByName('ProductName').AsString :=
        GetProductName(qyMaster.FieldByName('ProductID').AsString);
end;

procedure TfmIN100.qyMasterBeforeEdit(DataSet: TDataSet);
begin
  if (qyMaster.FieldByName('InventoryYear').AsString + '-' +
     qyMaster.FieldByName('InventoryMonth').AsString + '-01') <> sStartPeriodDate then
  begin
    MyWarning('Openning data not at the Openning month, cannot be edited    ');
    Abort;
  end;
  inherited;
end;
```

```
procedure TfmIN100.qyMasterBeforeInsert(DataSet: TDataSet);
begin
 if sOpenningDate <> sStartPeriodDate then
 begin
  MyWarning('Transaction period not at the openning month, new record not allow  ');
  Abort;
 end;
 inherited;
end;

procedure TfmIN100.sbSelectClick(Sender: TObject);
begin
 inherited;
 ReData := False;
 qyMaster.DisableControls;
 try
  fmIN100Select := TfmIN100Select.Create(Application);
  fmIN100Select.ShowModal;
  if ReData then begin
   try
    fmLoading := TfmLoading.Create(Application);
    fmLoading.Show;
    fmLoading.Update;
    qyMaster.Close;
    qyMaster.Open;
   finally
    fmLoading.Hide;
    fmLoading.Free;
   end;
  end;
 finally
  qyMaster.EnableControls;
  fmIN100Select.Free;
 end;
end;

end.
```




```
unit AP110;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  MasterDetail, Menus, Db, DBTables, Grids, DBGrids, Buttons, ExtCtrls,
  StdCtrls, Mask, DBCtrls;

type
  TfmAP110 = class(TfmMasterDetail)
    qyMasterCompanyID: TStringField;
    qyMasterDiscount: TBCDField;
    qyMasterRemittance: TBCDField;
    qyMasterOthers: TBCDField;
    qyMasterTotalBalance: TBCDField;
    qyMasterCreateMan: TStringField;
    qyMasterCreateDate: TStringField;
    qyMasterCreateTime: TStringField;
    qyMasterModifyMan: TStringField;
    qyMasterModifyDate: TStringField;
    qyMasterModifyTime: TStringField;
    Label1: TLabel;
```

```
DBEdit1: TDBEdit;
Label2: TLabel;
Label3: TLabel;
ED2: TDBEdit;
DBEdit3: TDBEdit;
Label5: TLabel;
ED3: TDBEdit;
Label6: TLabel;
ED4: TDBEdit;
Label7: TLabel;
ED5: TDBEdit;
Label8: TLabel;
ED6: TDBEdit;
Label9: TLabel;
ED7: TDBEdit;
Label10: TLabel;
ED8: TDBEdit;
Label11: TLabel;
DBEdit10: TDBEdit;
Label12: TLabel;
DBEdit11: TDBEdit;
sbSupplierID: TSpeedButton;
qyDetailBalance: TBCDField;
sbPayDate: TSpeedButton;
qyPurchaseMaster: TQuery;
edPurchaseID: TDBEdit;
qyMasterSupplierAttribName: TStringField;
qyMasterSupplierID: TStringField;
qyMasterPaymentID: TStringField;
qyMasterPayDate: TStringField;
qyMasterPayCash: TBCDField;
qyMasterPayCheck: TBCDField;
qyMasterPrepaid: TBCDField;
qyDetailPaymentID: TStringField;
qyDetailPurchaseID: TStringField;
qyDetailPaid: TBCDField;
qyDetailAccountPayable: TBCDField;
qyDetailUnPaid: TFloatField;
qyDetailCompanyID: TStringField;
qyPurchaseMasterPurchaseID: TStringField;
qyPurchaseMasterPurchaseDate: TStringField;
qyPurchaseMasterAccountPayable: TBCDField;
qyPurchaseMasterPaid: TBCDField;
qyMasterPayAmount: TBCDField;
procedure FormCreate(Sender: TObject);
procedure qyMasterBeforeOpen(DataSet: TDataSet);
```

```
    procedure qyMasterNewRecord(DataSet: TDataSet);
    procedure qyMasterBeforePost(DataSet: TDataSet);
    procedure qyDetailBeforeOpen(DataSet: TDataSet);
    procedure qyDetailNewRecord(DataSet: TDataSet);
    procedure qyDetailBeforePost(DataSet: TDataSet);
    procedure qyDetailCalcFields(DataSet: TDataSet);
    procedure sbPayDateClick(Sender: TObject);
    procedure sbSupplierIDClick(Sender: TObject);
    procedure dsMasterStateChange(Sender: TObject);
    procedure gdDetailEditButtonClick(Sender: TObject);
    procedure qyPurchaseMasterBeforeOpen(DataSet: TDataSet);
    procedure qyMasterUpdateRecord(DataSet: TDataSet;
      UpdateKind: TUpdateKind; var UpdateAction: TUpdateAction);
    procedure qyDetailUpdateRecord(DataSet: TDataSet;
      UpdateKind: TUpdateKind; var UpdateAction: TUpdateAction);
    procedure qyDetailBeforeInsert(DataSet: TDataSet);
    procedure qyDetailBeforeEdit(DataSet: TDataSet);
    procedure qyMasterSupplierIDValidate(Sender: TField);
    procedure qyDetailPurchaseIDValidate(Sender: TField);
    procedure sbSelectClick(Sender: TObject);
  private
    { Private declarations }
    procedure GetPurchaseInformation;
  public
    { Public declarations }
    AStartPeriodDate, AEndPeriodDate : String;
    AStartSupplierID, AEndSupplierID : String;
    ReData : Boolean;
    procedure OpenDB; override;
    procedure CalcAmount; override;
  end;

var
  fmAP110: TfmAP110;

implementation

uses Main, DataModule, PublicFunction, CheckData, GetData,
  BaseSearch, AP110Select, Loading, CalendarSearch;

var
  fTempAccountPayable, fTempPaid : Extended;
  sTempPurchaseID : String;
  fPrevBalance : Extended;

{$R *.DFM}
```

```
{ TfmAP110 }

procedure TfmAP110.FormCreate(Sender: TObject);
begin

  AStartPeriodDate := sStartPeriodDate;
  AEndPeriodDate := sEndPeriodDate;
  AStartSupplierID := '00000000';
  AEndSupplierID := 'ZZZZZZZZ';
  ReData := False;
  inherited;

  fTempAccountPayable := 0;
  fTempPaid := 0;
  sTempPurchaseID := '';
  fPrevBalance := 0;
end;

procedure TfmAP110.OpenDB;
begin
  qyMaster.Close;
  OrderBySQL := 'M.PaymentID';
  qyMaster.SQL.Text := SQLText;
  qyMaster.Open;

end;

procedure TfmAP110.CalcAmount;
var
  fBalance : Extended;
  PrevRecord : TBookmark;
begin

  PrevRecord := qyDetail.GetBookmark;
  qyDetail.DisableControls;
  try
    fBalance := 0;
    qyDetail.First;
    while not qyDetail.Eof do
    begin
      fBalance := fBalance + qyDetail.FieldByName('Balance').AsFloat;
      qyDetail.Next;
    end;
    qyMaster.FieldByName('TotalBalance').AsFloat := fBalance;
  finally
```

```
    qyDetail.EnableControls;
    if PrevRecord <> nil then
    begin
     qyDetail.GotoBookmark(PrevRecord);
     qyDetail.FreeBookmark(PrevRecord);
    end;
  end;
end;

procedure TfmAP110.qyMasterBeforeOpen(DataSet: TDataSet);
begin
  inherited;

  with qyMaster do
  begin
    ParamByName('StartPeriodDate').AsString := AStartPeriodDate;
    ParamByName('EndPeriodDate').AsString := AEndPeriodDate;
    ParamByName('StartSupplierID').AsString := AStartSupplierID;
    ParamByName('EndSupplierID').AsString := AEndSupplierID;
  end;
end;

procedure TfmAP110.qyMasterNewRecord(DataSet: TDataSet);
begin
  inherited;

  with qyMaster do
  begin
    FieldByName('PaymentID').AsString := Space(10);
    FieldByName('PayDate').AsString := Today;
    FieldByName('SupplierID').AsString := Space(10);
    FieldByName('PayCash').AsFloat := 0;
    FieldByName('PayCheck').AsFloat := 0;
    FieldByName('Discount').AsFloat := 0;
    FieldByName('Remittance').AsFloat := 0;
    FieldByName('Prepaid').AsFloat := 0;
    FieldByName('Others').AsFloat := 0;
    FieldByName('PayAmount').AsFloat := 0;
    FieldByName('TotalBalance').AsFloat := 0;
  end;
end;

procedure TfmAP110.qyMasterBeforePost(DataSet: TDataSet);
begin
  inherited;
```

```
 if qyMaster.State = dsInsert then
 begin
  if not CheckDate(qyMaster.FieldByName('PayDate').AsString) then
  begin
   ED1.SetFocus;
   Abort;
  end;
  if not CheckSupplierID(qyMaster.FieldByName('SupplierID').AsString) then
  begin
   ED2.SetFocus;
   Abort;
  end;
  qyMaster.FieldByName('PaymentID').AsString :=
       IDGen('AP', qyMaster.FieldByName('PayDate').AsString,
          'PaymentID', 'AccountPayableMaster');
 end;
 qyMaster.FieldByName('PayAmount').AsFloat :=
      qyMaster.FieldByName('PayCash').AsFloat +
      qyMaster.FieldByName('PayCheck').AsFloat +
      qyMaster.FieldByName('Discount').AsFloat +
      qyMaster.FieldByName('Remittance').AsFloat -
      qyMaster.FieldByName('Prepaid').AsFloat -
      qyMaster.FieldByName('Others').AsFloat;
end;

procedure TfmAP110.qyDetailBeforeOpen(DataSet: TDataSet);
begin
 inherited;

 qyDetail.ParamByName('PaymentID').AsString := qyMaster.FieldByName('PaymentID').AsString;
end;

procedure TfmAP110.qyDetailNewRecord(DataSet: TDataSet);
begin
 inherited;

 with qyDetail do
 begin
  FieldByName('PaymentID').AsString := qyMaster.FieldByName('PaymentID').AsString;
  FieldByName('PurchaseID').AsString := Space(10);
  FieldByName('Balance').AsFloat := 0;
 end;
end;

procedure TfmAP110.qyDetailBeforePost(DataSet: TDataSet);
var
```

```
  fPaid : Extended;
begin
  inherited;

  GetPurchaseInformation;
  if sTempPurchaseID = '' then
  begin
    NotFoundWarning('Purchases Invoice No.', qyDetail.FieldByName('PurchaseID').AsString);
    gdDetail.SelectedField := qyDetailPurchaseID;
    Abort;
  end;
  if fTempAccountPayable = fTempPaid then
  begin
    MyWarning('Purchase ID    [' +
          qyDetail.FieldByName('PurchaseID').AsString + ']' + #10#13 +
          'Knock off completed, please enter again    ');
    gdDetail.SelectedField := qyDetailPurchaseID;
    Abort;
  end;
  if qyDetail.FieldByName('Balance').AsFloat = 0 then
  begin
    MyWarning('This knock off cannot be 0, please re-enter    ');
    gdDetail.SelectedField := qyDetailBalance;
    Abort;
  end;
  fPaid := qyDetail.FieldByName('Paid').AsFloat +
          qyDetail.FieldByName('Balance').AsFloat -
          fPrevBalance;
  if ((fTempAccountPayable < 0) and (fPaid < fTempAccountPayable)) or
    ((fTempAccountPayable > 0) and (fPaid > fTempAccountPayable)) then
  begin
    MyWarning('This knock off cannot > knock off balance available, please re-enter    ');
    gdDetail.SelectedField := qyDetailBalance;
    Abort;
  end;
  qyDetail.FieldByName('Paid').AsFloat := fPaid;
end;

procedure TfmAP110.qyDetailCalcFields(DataSet: TDataSet);
begin
  inherited;

  qyDetail.FieldByName('UnPaid').AsFloat :=
        qyDetail.FieldByName('AccountPayable').AsFloat -
        qyDetail.FieldByName('Paid').AsFloat;
end;
```

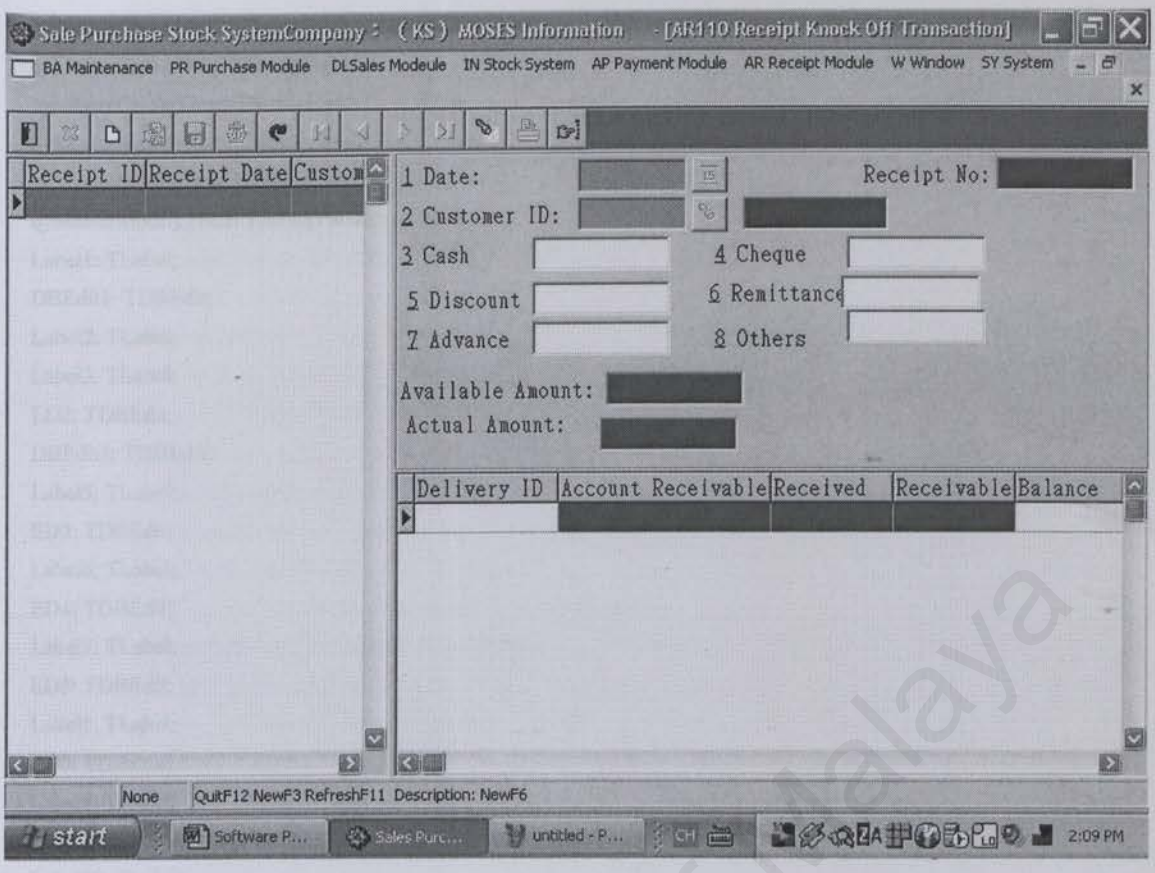
```
procedure TfmAP110.sbPayDateClick(Sender: TObject);
begin
  inherited;
// MyCalendar(ED1);
  ShowCalendar(ED1);
end;

procedure TfmAP110.sbSupplierIDClick(Sender: TObject);
begin
  inherited;
  SearchData(ED2, DM.qySupplier);
end;

procedure TfmAP110.dsMasterStateChange(Sender: TObject);
begin
  inherited;
  ED2.ReadOnly := (qyMaster.State <> dsInsert);
  sbPayDate.Enabled := (qyMaster.State = dsInsert);
  sbSupplierID.Enabled := (qyMaster.State = dsInsert);
end;

procedure TfmAP110.gdDetailEditButtonClick(Sender: TObject);
begin
  inherited;
  if (gdDetail.SelectedField.FieldName = 'PurchaseID') and
    (qyDetail.State in [dsInsert, dsEdit]) then
    qyDetail.FieldByName('PurchaseID').AsString := SearchData(edPurchaseID, qyPurchaseMaster);
end;

procedure TfmAP110.qyPurchaseMasterBeforeOpen(DataSet: TDataSet);
begin
  inherited;
  with qyPurchaseMaster do
  begin
    ParamByName('CompanyID').AsString := sCompanyID;
    ParamByName('SupplierID').AsString := qyMaster.FieldByName('SupplierID').AsString;
  end;
end;

procedure TfmAP110.qyMasterUpdateRecord(DataSet: TDataSet;
  UpdateKind: TUpdateKind; var UpdateAction: TUpdateAction);
var
  fOldPrepaid, fNewPrepaid : Extended;
  sSupplierID : String;
begin
```

```
 inherited;
 fOldPrepaid := 0;
 fNewPrepaid := 0;
 case UpdateKind of
  ukInsert :
   begin
    fNewPrepaid := DataSet.FieldByName('Prepaid').NewValue;
    fOldPrepaid := 0;
    sSupplierID := DataSet.FieldByName('SupplierID').NewValue;
   end;
  ukModify :
   begin
    fNewPrepaid := DataSet.FieldByName('Prepaid').NewValue;
    fOldPrepaid := DataSet.FieldByName('Prepaid').OldValue;
    sSupplierID := DataSet.FieldByName('SupplierID').OldValue;
   end;
  ukDelete :
   begin
    fNewPrepaid := DataSet.FieldByName('Prepaid').OldValue;
    fOldPrepaid := DataSet.FieldByName('Prepaid').OldValue;
    sSupplierID := DataSet.FieldByName('SupplierID').OldValue;
   end;
 end;
 with qyTemp do
 begin
  Close;
  SQL.Clear;
  SQL.Add('UPDATE Supplier SET Prepaid = Prepaid + :Prepaid ');
  SQL.Add('WHERE CompanyID = :CompanyID AND SupplierID = :SupplierID ');
  ParamByName('CompanyID').AsString := sCompanyID;
  ParamByName('SupplierID').AsString := sSupplierID;
  ParamByName('Prepaid').AsFloat := fNewPrepaid - fOldPrepaid;
  ExecSQL;
 end;
end;

procedure TfmAP110.qyDetailUpdateRecord(DataSet: TDataSet;
 UpdateKind: TUpdateKind; var UpdateAction: TUpdateAction);
 procedure UpdatePurchaseMasterPaid(PurchaseID: String; Paid: Extended);
 begin
  with qyTemp do
  begin
   Close;
   SQL.Clear;
   SQL.Add('UPDATE PurchaseMaster SET Paid = Paid + :Paid ');
   SQL.Add('WHERE CompanyID = :CompanyID AND PurchaseID = :PurchaseID ');
```

```
    ParamByName('CompanyID').AsString := sCompanyID;
    ParamByName('PurchaseID').AsString := PurchaseID;
    ParamByName('Paid').AsFloat := Paid;
    ExecSQL;
   end;
  end;
var
  fOldBalance, fNewBalance : Extended;
  fOldPurchaseID, fNewPurchaseID : String;
begin
  inherited;
  fNewBalance := 0;
  fOldBalance := 0;
  fNewPurchaseID := '';
  fOldPurchaseID := '';
  case UpdateKind of
   ukInsert :
    begin
     fNewBalance := DataSet.FieldByName('Balance').NewValue;
     fOldBalance := 0;
     fNewPurchaseID := DataSet.FieldByName('PurchaseID').NewValue;
     fOldPurchaseID := DataSet.FieldByName('PurchaseID').NewValue;
    end;
   ukModify :
    begin
     fNewBalance := DataSet.FieldByName('Balance').NewValue;
     fOldBalance := DataSet.FieldByName('Balance').OldValue;
     fNewPurchaseID := DataSet.FieldByName('PurchaseID').NewValue;
     fOldPurchaseID := DataSet.FieldByName('PurchaseID').OldValue;
    end;
   ukDelete :
    begin
     fNewBalance := 0;
     fOldBalance := DataSet.FieldByName('Balance').OldValue;
     fNewPurchaseID := DataSet.FieldByName('PurchaseID').OldValue;
     fOldPurchaseID := DataSet.FieldByName('PurchaseID').OldValue;
    end;
  end;
  UpdatePurchaseMasterPaid(fOldPurchaseID, fOldBalance * -1);
  UpdatePurchaseMasterPaid(fNewPurchaseID, fNewBalance);
end;

procedure TfmAP110.GetPurchaseInformation;
begin
  with qyTemp do
  begin
```

```
    Close;
    SQL.Clear;
    SQL.Add('SELECT PurchaseID, AccountPayable, Paid ');
    SQL.Add('FROM PurchaseMaster ');
    SQL.Add('WHERE CompanyID = :CompanyID AND PurchaseID = :PurchaseID ');
    SQL.Add('AND SupplierID = :SupplierID ');
    SQL.Add('AND (PurchaseProperty IN ("5", "6")) ');
    ParamByName('CompanyID').AsString := sCompanyID;
    ParamByName('PurchaseID').AsString := qyDetail.FieldByName('PurchaseID').AsString;
    ParamByName('SupplierID').AsString := qyMaster.FieldByName('SupplierID').AsString;
    Open;
  end;
  fTempAccountPayable := qyTemp.FieldByName('AccountPayable').AsFloat;
  fTempPaid := qyTemp.FieldByName('Paid').AsFloat;
  sTempPurchaseID := qyTemp.FieldByName('PurchaseID').AsString;
end;

procedure TfmAP110.qyDetailBeforeInsert(DataSet: TDataSet);
begin
  inherited;
  fPrevBalance := 0;
end;

procedure TfmAP110.qyDetailBeforeEdit(DataSet: TDataSet);
begin
  inherited;
  fPrevBalance := qyDetail.FieldByName('Balance').AsFloat;
end;

procedure TfmAP110.qyMasterSupplierIDValidate(Sender: TField);
begin
  inherited;
  qyMaster.FieldByName('SupplierAttribName').AsString :=
        GetSupplierAttribName(qyMaster.FieldByName('SupplierID').AsString);

end;

procedure TfmAP110.qyDetailPurchaseIDValidate(Sender: TField);
begin
  inherited;
  GetPurchaseInformation;
  qyDetail.FieldByName('Paid').AsFloat := fTempPaid;
  qyDetail.FieldByName('AccountPayable').AsFloat := fTempAccountPayable;
  qyDetail.FieldByName('Balance').AsFloat := fTempAccountPayable - fTempPaid;
end;
```

```
procedure TfmAP110.sbSelectClick(Sender: TObject);
begin
 inherited;
 ReData := False;
 qyMaster.DisableControls;
 try
  fmAP110Select := TfmAP110Select.Create(Application);
  fmAP110Select.ShowModal;
  if ReData then begin
   try
    fmLoading := TfmLoading.Create(Application);
    fmLoading.Show;
    fmLoading.Update;
    qyMaster.Close;
    qyMaster.Open;
   finally
    fmLoading.Hide;
    fmLoading.Free;
   end;
  end;
 finally
  qyMaster.EnableControls;
  fmAP110Select.Free;
 end;
end;

end.
```




```
unit AR110;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  MasterDetail, Menus, Db, DBTables, Grids, DBGrids, Buttons, ExtCtrls,
  StdCtrls, Mask, DBCtrls;

type
  TfmAR110 = class(TfmMasterDetail)
    qyMasterCustomerAttribName: TStringField;
    qyMasterCompanyID: TStringField;
    qyMasterCustomerID: TStringField;
    qyMasterReceiveID: TStringField;
    qyMasterReceiveDate: TStringField;
    qyMasterReceiveCash: TBCDField;
    qyMasterReceiveCheck: TBCDField;
    qyMasterDiscount: TBCDField;
    qyMasterRemittance: TBCDField;
    qyMasterAdvance: TBCDField;
    qyMasterOthers: TBCDField;
    qyMasterTotalBalance: TBCDField;
```

```
qyMasterCreateMan: TStringField;
qyMasterCreateDate: TStringField;
qyMasterCreateTime: TStringField;
qyMasterModifyMan: TStringField;
qyMasterModifyDate: TStringField;
qyMasterModifyTime: TStringField;
Label1: TLabel;
DBEdit1: TDBEdit;
Label2: TLabel;
Label3: TLabel;
ED2: TDBEdit;
DBEdit3: TDBEdit;
Label5: TLabel;
ED3: TDBEdit;
Label6: TLabel;
ED4: TDBEdit;
Label7: TLabel;
ED5: TDBEdit;
Label8: TLabel;
ED6: TDBEdit;
Label9: TLabel;
ED7: TDBEdit;
Label10: TLabel;
ED8: TDBEdit;
Label11: TLabel;
DBEdit10: TDBEdit;
Label12: TLabel;
DBEdit11: TDBEdit;
sbCustomerID: TSpeedButton;
qyDetailCompanyID: TStringField;
qyDetailReceiveID: TStringField;
qyDetailDeliveryID: TStringField;
qyDetailBalance: TBCDField;
qyDetailReceived: TBCDField;
qyDetailUnReceived: TFloatField;
sbReceiveDate: TSpeedButton;
qyDeliveryMaster: TQuery;
edDeliveryID: TDBEdit;
qyMasterReceiveAmount: TBCDField;
qyDeliveryMasterDeliveryID: TStringField;
qyDeliveryMasterDeliveryDate: TStringField;
qyDeliveryMasterReceived: TBCDField;
qyDeliveryMasterAccountReceivable: TBCDField;
qyDetailAccountReceivable: TBCDField;
procedure FormCreate(Sender: TObject);
procedure qyMasterBeforeOpen(DataSet: TDataSet);
```

```
    procedure qyMasterNewRecord(DataSet: TDataSet);
    procedure qyMasterBeforePost(DataSet: TDataSet);
    procedure qyDetailBeforeOpen(DataSet: TDataSet);
    procedure qyDetailNewRecord(DataSet: TDataSet);
    procedure qyDetailBeforePost(DataSet: TDataSet);
    procedure qyDetailCalcFields(DataSet: TDataSet);
    procedure qyMasterCustomerIDValidate(Sender: TField);
    procedure sbReceiveDateClick(Sender: TObject);
    procedure sbCustomerIDClick(Sender: TObject);
    procedure dsMasterStateChange(Sender: TObject);
    procedure gdDetailEditButtonClick(Sender: TObject);
    procedure qyDeliveryMasterBeforeOpen(DataSet: TDataSet);
    procedure qyMasterUpdateRecord(DataSet: TDataSet;
      UpdateKind: TUpdateKind; var UpdateAction: TUpdateAction);
    procedure qyDetailUpdateRecord(DataSet: TDataSet;
      UpdateKind: TUpdateKind; var UpdateAction: TUpdateAction);
    procedure qyDetailDeliveryIDValidate(Sender: TField);
    procedure qyDetailBeforeInsert(DataSet: TDataSet);
    procedure qyDetailBeforeEdit(DataSet: TDataSet);
    procedure sbSelectClick(Sender: TObject);
  private
    { Private declarations }
    procedure GetDeliveryInformation;
  public
    { Public declarations }
    AStartPeriodDate, AEndPeriodDate : String;
    AStartCustomerID, AEndCustomerID : String;
    ReData : Boolean;
    procedure OpenDB; override;
    procedure CalcAmount; override;
  end;

var
  fmAR110: TfmAR110;

implementation

uses Main, DataModule, PublicFunction, CheckData, GetData,
  BaseSearch, AR110Select, Loading, CalendarSearch;

var
  fTempAccountReceivable, fTempReceived : Extended;
  sTempDeliveryID : String;
  fPrevBalance : Extended;

{$R *.DFM}
```

```
{ TfmAR110 }

procedure TfmAR110.FormCreate(Sender: TObject);
begin

  AStartPeriodDate := sStartPeriodDate;
  AEndPeriodDate := sEndPeriodDate;
  AStartCustomerID := '00000000';
  AEndCustomerID := 'ZZZZZZZZ';
  ReData := False;
  inherited;

  fTempAccountReceivable := 0;
  fTempReceived := 0;
  sTempDeliveryID := '';
  fPrevBalance := 0;
end;

procedure TfmAR110.OpenDB;
begin
  qyMaster.Close;
  OrderBySQL := 'M.ReceiveID';
  qyMaster.SQL.Text := SQLText;
  qyMaster.Open;

end;

procedure TfmAR110.CalcAmount;
var
  fBalance : Extended;
  PrevRecord : TBookmark;
begin

  PrevRecord := qyDetail.GetBookmark;
  qyDetail.DisableControls;
  try
    fBalance := 0;
    qyDetail.First;
    while not qyDetail.Eof do
    begin
      fBalance := fBalance + qyDetail.FieldByName('Balance').AsFloat;
      qyDetail.Next;
    end;
    qyMaster.FieldByName('TotalBalance').AsFloat := fBalance;
  finally
```

```
    qyDetail.EnableControls;
    if PrevRecord <> nil then
    begin
     qyDetail.GotoBookmark(PrevRecord);
     qyDetail.FreeBookmark(PrevRecord);
    end;
  end;
end;

procedure TfmAR110.qyMasterBeforeOpen(DataSet: TDataSet);
begin
  inherited;

  with qyMaster do
  begin
    ParamByName('StartPeriodDate').AsString := AStartPeriodDate;
    ParamByName('EndPeriodDate').AsString := AEndPeriodDate;
    ParamByName('StartCustomerID').AsString := AStartCustomerID;
    ParamByName('EndCustomerID').AsString := AEndCustomerID;
  end;
end;

procedure TfmAR110.qyMasterNewRecord(DataSet: TDataSet);
begin
  inherited;

  with qyMaster do
  begin
    FieldByName('ReceiveID').AsString := Space(10);
    FieldByName('ReceiveDate').AsString := Today;
    FieldByName('CustomerID').AsString := Space(10);
    FieldByName('ReceiveCash').AsFloat := 0;
    FieldByName('ReceiveCheck').AsFloat := 0;
    FieldByName('Discount').AsFloat := 0;
    FieldByName('Remittance').AsFloat := 0;
    FieldByName('Advance').AsFloat := 0;
    FieldByName('Others').AsFloat := 0;
    FieldByName('ReceiveAmount').AsFloat := 0;
    FieldByName('TotalBalance').AsFloat := 0;
  end;
end;

procedure TfmAR110.qyMasterBeforePost(DataSet: TDataSet);
begin
  inherited;
```

```
 if qyMaster.State = dsInsert then
 begin
  if not CheckDate(qyMaster.FieldByName('ReceiveDate').AsString) then
  begin
   ED1.SetFocus;
   Abort;
  end;
  if not CheckCustomerID(qyMaster.FieldByName('CustomerID').AsString) then
  begin
   ED2.SetFocus;
   Abort;
  end;
  qyMaster.FieldByName('ReceiveID').AsString :=
       IDGen('AR', qyMaster.FieldByName('ReceiveDate').AsString,
          'ReceiveID', 'AccountReceivableMaster');
 end;
 qyMaster.FieldByName('ReceiveAmount').AsFloat :=
      qyMaster.FieldByName('ReceiveCash').AsFloat +
      qyMaster.FieldByName('ReceiveCheck').AsFloat +
      qyMaster.FieldByName('Discount').AsFloat +
      qyMaster.FieldByName('Remittance').AsFloat -
      qyMaster.FieldByName('Advance').AsFloat -
      qyMaster.FieldByName('Others').AsFloat;
end;

procedure TfmAR110.qyDetailBeforeOpen(DataSet: TDataSet);
begin
 inherited;

 qyDetail.ParamByName('ReceiveID').AsString := qyMaster.FieldByName('ReceiveID').AsString;
end;

procedure TfmAR110.qyDetailNewRecord(DataSet: TDataSet);
begin
 inherited;

 with qyDetail do
 begin
  FieldByName('ReceiveID').AsString := qyMaster.FieldByName('ReceiveID').AsString;
  FieldByName('DeliveryID').AsString := Space(10);
  FieldByName('Balance').AsFloat := 0;
 end;
end;

procedure TfmAR110.qyDetailBeforePost(DataSet: TDataSet);
var
```

```
 fReceived : Extended;
begin
 inherited;

 GetDeliveryInformation;
 if sTempDeliveryID = '' then
 begin
  NotFoundWarning('Sales Invoice No.', qyDetail.FieldByName('DeliveryID').AsString);
  gdDetail.SelectedField := qyDetailDeliveryID;
  Abort;
 end;
 if fTempAccountReceivable = fTempReceived then
 begin
  MyWarning('Delivery ID    [' +
        qyDetail.FieldByName('DeliveryID').AsString + ']' + #10#13 +
        'knock off completed, pleas enter again    ');
  gdDetail.SelectedField := qyDetailDeliveryID;
  Abort;
 end;
 if qyDetail.FieldByName('Balance').AsFloat = 0 then
 begin
  MyWarning('This knock off cannot be 0, please re-enter    ');
  gdDetail.SelectedField := qyDetailBalance;
  Abort;
 end;
 fReceived := qyDetail.FieldByName('Received').AsFloat +
        qyDetail.FieldByName('Balance').AsFloat -
        fPrevBalance;
 if ((fTempAccountReceivable < 0) and (fReceived < fTempAccountReceivable)) or
   ((fTempAccountReceivable > 0) and (fReceived > fTempAccountReceivable)) then
 begin
  MyWarning('This knock off cannot > knock off balance available, please re-enter    ');
  gdDetail.SelectedField := qyDetailBalance;
  Abort;
 end;
 qyDetail.FieldByName('Received').AsFloat := fReceived;
end;

procedure TfmAR110.qyDetailCalcFields(DataSet: TDataSet);
begin
 inherited;

 qyDetail.FieldByName('UnReceived').AsFloat :=
       qyDetail.FieldByName('AccountReceivable').AsFloat -
       qyDetail.FieldByName('Received').AsFloat;
end;
```

```
procedure TfmAR110.qyMasterCustomerIDValidate(Sender: TField);
begin
  inherited;
  qyMaster.FieldByName('CustomerAttribName').AsString :=
      GetCustomerAttribName(qyMaster.FieldByName('CustomerID').AsString);
end;

procedure TfmAR110.sbReceiveDateClick(Sender: TObject);
begin
  inherited;
//  MyCalendar(ED1);
  ShowCalendar(ED1);
end;

procedure TfmAR110.sbCustomerIDClick(Sender: TObject);
begin
  inherited;
  SearchData(ED2, DM.qyCustomerMaster);
end;

procedure TfmAR110.dsMasterStateChange(Sender: TObject);
begin
  inherited;
  ED2.ReadOnly := (qyMaster.State <> dsInsert);
  sbReceiveDate.Enabled := (qyMaster.State = dsInsert);
  sbCustomerID.Enabled := (qyMaster.State = dsInsert);
end;

procedure TfmAR110.gdDetailEditButtonClick(Sender: TObject);
begin
  inherited;
  if (gdDetail.SelectedField.FieldName = 'DeliveryID') and
    (qyDetail.State in [dsInsert, dsEdit]) then
    qyDetail.FieldByName('DeliveryID').AsString := SearchData(edDeliveryID, qyDeliveryMaster);
end;

procedure TfmAR110.qyDeliveryMasterBeforeOpen(DataSet: TDataSet);
begin
  inherited;
  with qyDeliveryMaster do
  begin
    ParamByName('CompanyID').AsString := sCompanyID;
    ParamByName('CustomerID').AsString := qyMaster.FieldByName('CustomerID').AsString;
  end;
end;
```

```
procedure TfmAR110.qyMasterUpdateRecord(DataSet: TDataSet;
  UpdateKind: TUpdateKind; var UpdateAction: TUpdateAction);
var
  fOldAdvance, fNewAdvance : Extended;
  fOldTotalBalance, fNewTotalBalance : Extended;
  sCustomerID : String;
begin
  inherited;
  fNewAdvance := 0;
  fOldAdvance := 0;
  fNewTotalBalance := 0;
  fOldTotalBalance := 0;
  sCustomerID := '';
  case UpdateKind of
    ukInsert :
      begin
        fNewAdvance := DataSet.FieldByName('Advance').NewValue;
        fOldAdvance := 0;
        fNewTotalBalance := DataSet.FieldByName('TotalBalance').NewValue;
        fOldTotalBalance := 0;
        sCustomerID := DataSet.FieldByName('CustomerID').NewValue;
      end;
    ukModify :
      begin
        fNewAdvance := DataSet.FieldByName('Advance').NewValue;
        fOldAdvance := DataSet.FieldByName('Advance').OldValue;
        fNewTotalBalance := DataSet.FieldByName('TotalBalance').NewValue;
        fOldTotalBalance := DataSet.FieldByName('TotalBalance').OldValue;
        sCustomerID := DataSet.FieldByName('CustomerID').OldValue;
      end;
    ukDelete :
      begin
        fNewAdvance := DataSet.FieldByName('Advance').OldValue;
        fOldAdvance := DataSet.FieldByName('Advance').OldValue;
        fNewTotalBalance := DataSet.FieldByName('TotalBalance').OldValue;
        fOldTotalBalance := DataSet.FieldByName('TotalBalance').OldValue;
        sCustomerID := DataSet.FieldByName('CustomerID').OldValue;
      end;
  end;
  with qyTemp do
  begin
    Close;
    SQL.Clear;
    SQL.Add('UPDATE CustomerMaster SET Advance = Advance + :Advance, ');
    SQL.Add('CreditBalance = CreditBalance + :CreditBalance ');
```

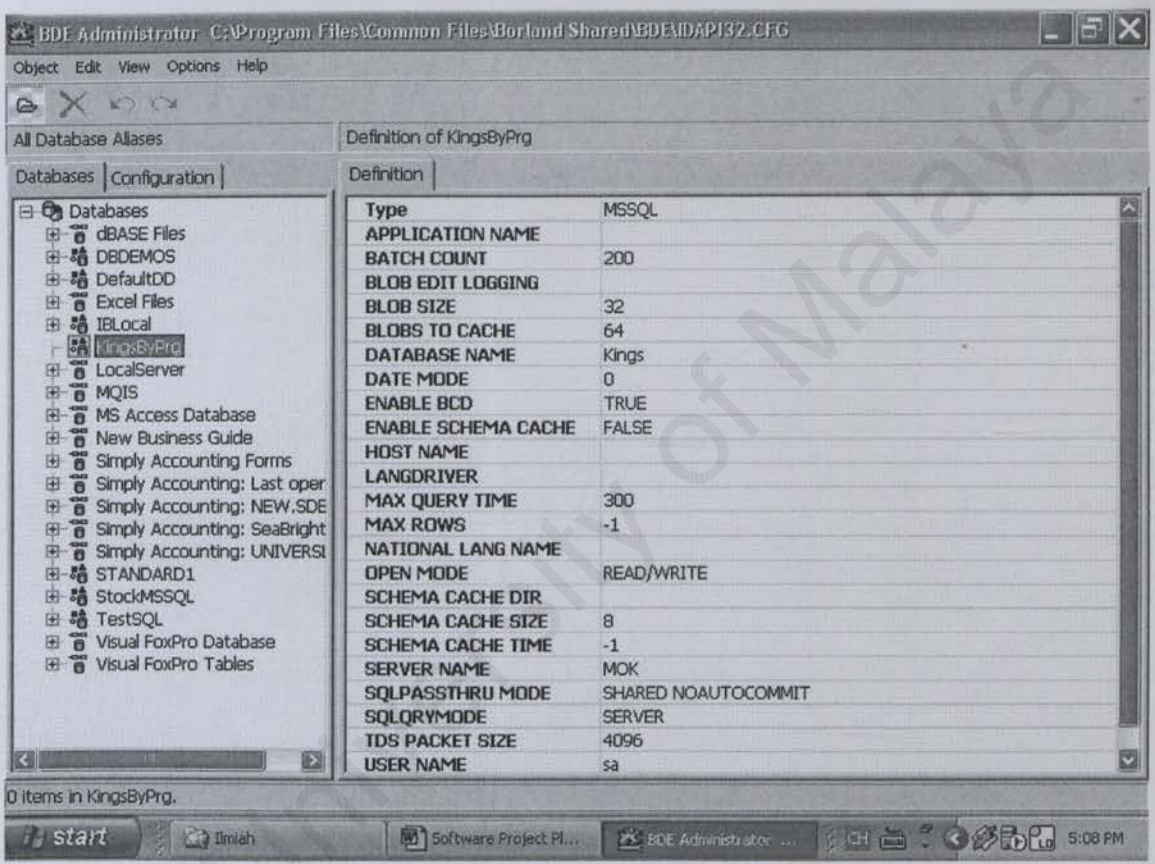
```
    SQL.Add('WHERE CompanyID = :CompanyID AND CustomerID = :CustomerID ');
    ParamByName('CompanyID').AsString := sCompanyID;
    ParamByName('CustomerID').AsString := sCustomerID;
    ParamByName('Advance').AsFloat := fNewAdvance - fOldAdvance;
    ParamByName('CreditBalance').AsFloat := fNewTotalBalance - fOldTotalBalance;
    ExecSQL;
  end;
end;

procedure TfmAR110.qyDetailUpdateRecord(DataSet: TDataSet;
  UpdateKind: TUpdateKind; var UpdateAction: TUpdateAction);
  procedure UpdateDeliveryMasterReceived(DeliveryID: String; Received: Extended);
  begin
    with qyTemp do
    begin
      Close;
      SQL.Clear;
      SQL.Add('UPDATE DeliveryMaster SET Received = Received + :Received ');
      SQL.Add('WHERE CompanyID = :CompanyID AND DeliveryID = :DeliveryID ');
      ParamByName('CompanyID').AsString := sCompanyID;
      ParamByName('DeliveryID').AsString := DeliveryID;
      ParamByName('Received').AsFloat := Received;
      ExecSQL;
    end;
  end;
var
  fOldBalance, fNewBalance : Extended;
  fOldDeliveryID, fNewDeliveryID : String;
begin
  inherited;
  fNewBalance := 0;
  fOldBalance := 0;
  fNewDeliveryID := '';
  fOldDeliveryID := '';
  case UpdateKind of
    ukInsert :
      begin
        fNewBalance := DataSet.FieldByName('Balance').NewValue;
        fOldBalance := 0;
        fNewDeliveryID := DataSet.FieldByName('DeliveryID').NewValue;
        fOldDeliveryID := DataSet.FieldByName('DeliveryID').NewValue;
      end;
    ukModify :
      begin
        fNewBalance := DataSet.FieldByName('Balance').NewValue;
        fOldBalance := DataSet.FieldByName('Balance').OldValue;
```

```
      fNewDeliveryID := DataSet.FieldByName('DeliveryID').NewValue;
      fOldDeliveryID := DataSet.FieldByName('DeliveryID').OldValue;
     end;
    ukDelete :
     begin
      fNewBalance := 0;
      fOldBalance := DataSet.FieldByName('Balance').OldValue;
      fNewDeliveryID := DataSet.FieldByName('DeliveryID').OldValue;
      fOldDeliveryID := DataSet.FieldByName('DeliveryID').OldValue;
     end;
  end;
  UpdateDeliveryMasterReceived(fOldDeliveryID, fOldBalance * -1);
  UpdateDeliveryMasterReceived(fNewDeliveryID, fNewBalance);
end;

procedure TfmAR110.qyDetailDeliveryIDValidate(Sender: TField);
begin
  inherited;
  GetDeliveryInformation;
  qyDetail.FieldByName('Received').AsFloat := fTempReceived;
  qyDetail.FieldByName('AccountReceivable').AsFloat := fTempAccountReceivable;
  qyDetail.FieldByName('Balance').AsFloat := fTempAccountReceivable - fTempReceived;
end;

procedure TfmAR110.GetDeliveryInformation;
begin
  with qyTemp do
  begin
    Close;
    SQL.Clear;
    SQL.Add('SELECT DeliveryID, AccountReceivable, Received ');
    SQL.Add('FROM DeliveryMaster ');
    SQL.Add('WHERE CompanyID = :CompanyID AND DeliveryID = :DeliveryID ');
    SQL.Add('AND CustomerID = :CustomerID ');
    SQL.Add('AND (DeliveryProperty IN ("1", "2")) ');
    ParamByName('CompanyID').AsString := sCompanyID;
    ParamByName('DeliveryID').AsString := qyDetail.FieldByName('DeliveryID').AsString;
    ParamByName('CustomerID').AsString := qyMaster.FieldByName('CustomerID').AsString;
    Open;
  end;
  fTempAccountReceivable := qyTemp.FieldByName('AccountReceivable').AsFloat;
  fTempReceived := qyTemp.FieldByName('Received').AsFloat;
  sTempDeliveryID := qyTemp.FieldByName('DeliveryID').AsString;
end;

procedure TfmAR110.qyDetailBeforeInsert(DataSet: TDataSet);
```

```
begin
  inherited;
  fPrevBalance := 0;
end;

procedure TfmAR110.qyDetailBeforeEdit(DataSet: TDataSet);
begin
  inherited;
  fPrevBalance := qyDetail.FieldByName('Balance').AsFloat;
end;

procedure TfmAR110.sbSelectClick(Sender: TObject);
begin
  inherited;
  ReData := False;
  qyMaster.DisableControls;
  try
    fmAR110Select := TfmAR110Select.Create(Application);
    fmAR110Select.ShowModal;
    if ReData then begin
      try
        fmLoading := TfmLoading.Create(Application);
        fmLoading.Show;
        fmLoading.Update;
        qyMaster.Close;
        qyMaster.Open;
      finally
        fmLoading.Hide;
        fmLoading.Free;
      end;
    end;
  finally
    qyMaster.EnableControls;
    fmAR110Select.Free;
  end;
end;

end.
```

**Installation Guide**

1. Install the Delphi 6 and MS SQL 7
2. Setup a local server in the OS system, the name used for the server is important database connectivity
3. Setup a databases in MS SQL, name as "Kings"
4. Copy databases of Sales, Purchases & Stock System from CD into the directory of MS SQL to replace the "Kings" databases created
5. Performed the databases connectivity with BDE Admistrtor of Delphi and amend the setting as the following figure:




6. Login procedures

   User ID : supervisor

   Password : 123

   Company : KS